Министерство образования Республики Беларусь

Учреждение образования
«Гомельский государственный университет
имени Франциска Скорины»

М.В.КОРОТКАЯ, Г.Н.ПЕТУХОВА, И.А.ХОРСУН,
Е.А.ЧЕРНЯКОВА

АНГЛИЙСКИЙ ЯЗЫК

ПРАКТИЧЕСКОЕ РУКОВОДСТВО

для студентов специальности 1-40 01 01 «Программное обеспечение информационных технологий»

Гомель 2013

УДК 811.111(075.8)
ББК 81.432.1-923
К 305

Рецензенты:
Пузенко И.Н. – заведующий кафедрой белорусского и иностранных языков учреждения образования «Гомельский государственный технический университет имени П.О.Сухого», кандидат филологических наук, доцент;
кафедра английского языка учреждения образования «Гомельский государственный университет имени Франциска Скорины»

Рекомендовано к изданию на заседании научно-методическим советом учреждения образования «Гомельский государственный университет имени Франциска Скорины».

М.В.Короткая, Г.Н.Петухова, И.А.Хорсун, Е.А.Чернякова

К 305 Английский язык: практическое руководство для студентов специальности 1-40 01 01 «Программное обеспечение информационных технологий», изучающих английский язык факультативно / М.В.Короткая, Г.Н.Петухова, И.А.Хорсун, Е.А.Чернякова; М-во образов РБ, Гомельский гос. ун-т им. Ф.Скорины. – Гомель: ГГУ им.Ф.Скорины, 2013 – 45 с.

ISBN 978-985-439-327-8

Целью практического руководства является оказание помощи студентам математического факультета, изучающих английский язык на факультативных занятиях, в накоплении и систематизации словарного запаса профессиональной лексики по предлагаемым темам и развитии навыков устной речи.

УДК 811.111(075.8)
ББК 81.432.1-923

ISBN 978-985-439-327-8

СОДЕРЖАНИЕ

ВВЕДЕНИЕ

Практическое руководство предназначено для студентов специальности «Программное обеспечение информационных технологий», изучающих английский язык на факультативных занятиях, и является неотъемлемой частью обучения иностранному языку. Оно способствует повышению уровня подготовки студентов по специальности и углубляет их заинтересованность в практическом овладении иностранным языком. В процессе обучения чтению совершенствуются навыки самостоятельного чтения и анализа текста, навыки работы со словарем и увеличивается словарный запас студентов.

Поскольку факультативное обучение это творческий процесс, авторы стремились подобрать разнообразный материал, который предполагает исследовательский процесс и укрепляет активную позицию студентов при обучении. Тексты уроков рассчитаны на специалистов в области информационных технологий. При их отборе авторы стремились к тому, чтобы они были разнообразны по структуре, информативны, познавательны. Кроме того, они построены на знакомых грамматических структурах, насыщены лексикой, связанной со специальностью, и легко пересказываются.

Объем текстов различен. По мере изучения материала меняется тематика и увеличивается объем текстов.

Каждый урок содержит новые лексические единицы и интернациональные слова, подлежащие активному усвоению. В состав текстов входит лексика, которая не изучалась по обязательной программе. Это исходные формы незнакомых слов в их основном значении и слова, которые трудно раскрыть обычным анализом словообразовательных элементов.

Послетекстовые упражнения разнообразны и соответствуют видам чтения (на понимание прочитанного, просмотровое, изучающее). Такие упражнения предназначены для обучения извлечению информации и развитию речевых навыков.

LESSON 1

Words to be Learned

Nouns: miracle, male/female, survey, statement, praise, item, gimmick, attitude (to), concern, score, addict, quest, access (to), overload.

Verbs: to deliver, to conscript, to spread, to overtake, to mess, to appreciate, to earn, to interfere, to complicate, to proliferate, to curtail, to confess to.

Adjectives: competitive, ambitious, vague, sensible, virtual, contemporary, brand-new, up-to-date, out-of-date, online.

Word combinations: vintage car, catch phrase, to surf the Internet, to be suspicious, to leave behind, to get frustrated.

Exercise 1

Read the text and analyze it

Computer studies?

1 If you're female, you're going to read this article and feel smug. If you are male, you might feel a desire to use the article to wrap up your old chewing gum or just get annoyed and play a computer game.

2 According to a recent report, in Britain girls are overtaking boys at school. They are even beating them in subjects such as science and maths, which people used to think, were subjects that boys were naturally better at. Surveys show there could be several reasons for this. Boys and girls behave very differently from each other both in and out of school.

3 In school, statistics show boys mess about more and get into trouble more. Admittedly, they put up their hand to answer questions more but they often have the wrong answer. The girls who were interviewed said they often knew the correct answer but didn't like to put up their hand if they weren't absolutely sure. The survey also showed girls spent much longer doing homework and checking it with each other. Boys may argue that these things do not make girls more intelligent than boys and in some boys' opinions may even make many girls look like swots. However, these things do show that girls have a different attitude to school than boys. Girls are becoming much more competitive and ambitious.

4 Experts believe that some boys are spending so much time pitying computer games and watching violent films that they are living in a fantasy world. When girls talk about using home computers, they often discuss different types of software that they use for learning. Boys simply talk about computer games. When 14-year-old girls were asked what they would like to do in the future, they mentioned realistic jobs such as vet, teacher or doctor. The boys' answers were either very vague such as, "I just want to be happy and have lots of money" or unrealistic and they said things such as, "I want to be a fighter pilot." Their answers were considered worrying because they did not seem very sensible and did not show any concern about unemployment. However, some people might believe that J4 is too young to worry anyhow. Also, the truth is that the majority of "top jobs" in England are still done by men so many might not see the need to worry. The good news is that after the age of 17, many boys become interested in school again and their exam results show that they have caught up. The problem is just keeping them interested until then...

5 A lot of knowledge is a dangerous thing for addicts of the Internet. Information is becoming the drug of the new century.

6 The research, conducted among 1000 managers in Britain, America, Europe and the Far East shows that, as information sources such as the Internet and cable news channels proliferate, we are witnessing the rise of a generation of dataholics.

7 The quest for information can lead to stress. Almost two-thirds said their leisure time had been curtailed as a result of having to work late to cope with vast amounts of information; 70 percent reported loss of job satisfaction and tension with colleagues because of information overload.

8 The study also investigated the habits of the children of 300 managers and found 55 per cent of parents were concerned their children would become information junkies.

9 Forty-six per cent of parents believed their children spent more time on their PCs than interacting with friends. In one case a child had to be wheeled with his computer to the dinner table.

10 Sue Feldman, mother of Alexander, 13, a self-confessed Internet-addict, said she had not yet been forced to wheeling her son

and computer to the table, but said she often served him sandwiches and crisps at his bedroom computer.

11 Alexander switches on his computer every day when he returns from Latymer School in Hammersmith to his home at Ealing, west London. "I'd confess to spending up to four hours a day on the Internet looking for information and speaking to friends. It's like an addiction," Alexander said.

12 "If I can't get on to my computer or the Internet, I do get really frustrated." He spends most of his time finding out the latest information on pop groups and facts for his homework.

13 "My parents have to tell me to get off the computer, and they complain a lot, but they also see it as a good thing. Practically everyone in my class has a PC with Internet access so all my friends are also on-line. It's the way forward."

Exercise 2

I *The statements below were other results of the survey. Write G if you think the statement might refer to girls and A if you think it could refer to boys.*

1 Learn to speak earlier.

2 Get nervous if there is a pause in the conversation between friends.

3 Take more risks.

4 Are spoken to more by parents.

5 Normally get more praise at school if they do something well.

6 Smoke more.

II *How modern are you? (pop quiz)*

1 If you were able to have any car you wanted, what would you buy?

a) I'd buy a restored vintage car that might become a collector's item.

b) I'd buy a newly built car with all the latest technology.

c) 1 wouldn't buy a car because 1 don't like them.

2 What is your attitude to new scientific developments?

a) They're brilliant. They will help to make the world a much happier and better place.

b) We know enough about science now. We should stop interfering with nature.

c) Some things are good. Some things are bad.

3 How do you speak?

a) 1 use a lot of new words, slang and catch phrases from the television and magazines.

b) I use exactly the same words and phrases as my parents.

c) I use a few new words because they are useful for what I want to say.

4 Which of the following do you think is the most enjoyable?

a) Playing virtual reality computer games.

b) Going to a disco club that plays music from the 60s and 70s.

c) Listening to techno music.

5 Which of the following would be your preferred way of finding out information?

a) I like looking it up in a book.

b) Surfing the Internet or using a CD-Rom is the best way.

c) Watching a video is best.

6 You go to a friend's house. Their mother earns a lot of money and works and their father stays at home, cooks and cleans. What is your reaction?

a) Nothing, it doesn't matter who works and who cleans. It's the 90s.

b) A bit surprised. It seems a bit strange because it's unusual,

c) The poor man. Cooking and cleaning is a woman's job.

7 Which of the following types of books or films do you prefer?

a) Historical ones,

b) Anything romantic.

c) Contemporary ones about modern day things.

8 If your computer was six years old and worked perfectly well, which of the following would you do?

a) I'd buy a brand new one so I could have new technology

b) I wouldn't do anything. I'd be happy with it. New technology is just gimmicks.

c) I'd secretly hope it would break, despite the fact that didn't need a new computer.

Add up your score and read the analysis

	a	b	c

1	2	3	1
2	3	1	2
3	3	1	2
4	3	1	2
5	1	3	2
6	3	2	1
7	1	2	3
8	3	1	2

The analysis

8-11: You are not modern at all and you don't want to be. You are suspicious of new things and don't make an effort to find out about hem. You would prefer to live in the past. It's nice that you can appreciate the simple things in life but you must be careful not to get left behind. You are too traditional.

12-16: You are not very modern but you are not completely old-fashioned either. You like to live in a world that has the good things from he past and some of the good things from the present too.

17-20: You are modern. You know a lot about what is happening around you and obviously enjoy progress. On the other hand, you are sensible and don't worry about buying and doing all the latest things because they are fashionable.

21-24: Yes. You are very modern. Being up-to-date is very important to you. Sometimes perhaps it is too important. Remember that new things are not always the best things. Be careful not to become obsessed vim every new thing that comes along. Some things are *just* clever marketing crazes that will complicate your life.

LESSON 2

Words to be Learned

Unhealthy, harmless, a padded chair, eyestrain, injury, exposure, ergonomics, wrists, a glare deflector, a tilt-and-swivel base, inexpensive.

Exercise 1

Read the text and analyze it

To your health

1 Can all this computing be good for you? Art there any unhealthy side effects? The computer seems harmless enough. How bad can it be, sitting in a padded chair in a climate-controlled office?

2 Health questions have been raised by the people who sit all day in front of the video display terminals (VDTs) of their computers. Are computer users getting bad radiation? What about eyestrain? And what about the age-old back problem, updated with new concerns about workers who hold their hands over a keyboard? What about repetitive-action injury also known as carpal tunnel syndrome? What about the risk of miscarriage?

3 Unions and legislators in many *communities continue to* push for laws limiting exposure to video screens. Many manufacturers now offer screens with built-in protection.

4 Meanwhile, there are a number of things workers can do to take care of themselves. A good place to begin is with an ergonomically designed workstation. Ergonomics is the study of human factors related to computers. A properly designed workstation takes a variety of factors into account, such as the distance from the eyes to the screen and the angle of the arms and wrists.

5 Experts recommend these steps as coping mechanisms:

– Turn the screen away from the window to reduce glare, and cover your screen with a glare deflector. Turn off overhead lights; illuminate your work area with a lamp.

– Put your monitor on a tilt-and-swivel base.

– Get a pneumatically adjustable chair. Position the seat back so your lower back is supported.

– Place the keyboard low enough to avoid arm and wrist fatigue. Do not bend your wrists when you type. Use an inexpensive, raised wrist rest. Do not rest your wrists on a sharp edge.

– Sit with your feet firmly on the floor.

– Exercise at your desk occasionally rotating your wrist, rolling your shoulders, and stretching. Better yet, get up and walk around at regular intervals.

Exercise 2

a. *Find in the text equivalents to*:

вредные побочные эффекты; мягкое кресло; вентилируемое помещение; электромагнитное излучение; облучение; радикулит; усталости рук и кистей; рабочее место; верхнее освещение; учитывать (принимать в расчет); наклонная и вращающаяся подставка; уменьшить свечение; встроенная защита; пневматически регулируемое кресло; вращать; потягиваться; острый край, регулярно.

b. *Fill in the table*:

Problem	Disease	How to cope
VDT radiation	Eyestrain, headache, immune system diseases, risk of miscarriage	Increase distance from the eyes to the screen Install radiation protection devices (a glare reflector)
Staying indoors		
Autism		

LESSON 3

Words to be Learned

flake, outstrip, miraculous device, unerring accountant, accomplished by, flip on and off, catatonia, incantation, natty, preternatural, nausea, sweaty palms, spinning reels, intimidating, glitch, havoc, foul up, issue.

Exercise 1

Read the text and analyze it

Worry about computers? me?

1 When your computer is turned off, it is a dead collection of sheet metal, plastic, metallic tracings, and tiny flakes of silicon. When you hit on switch, one little burst of electricity – only about 5 volts – starts a string of events that magically brings to life what otherwise would remain an oversize paperweight.

2 At first the PC is still rather stupid. Beyond taking inventory of itself, the newly awakened PC still can't do anything really useful, intelligent. At best it can search for intelligence in the form of

operating system that gives structure to the PC's primitive existence. Then comes a true education in the form of application software – programs that tell it how to do tasks faster and more accurately than we could, a student who has outstripped its teacher.

3 What makes your PC such a miraculous device is that each time' you turn it on, it is a tabula rasa, capable of doing anything your creativity – or, more usually, the creativity of professional programmers – can imagine for it to do. It is a calculating machine, a magical typewriter, an unerring accountant, and a host of other tools. To transform it from one persona to another requires setting some of the microscopic switches buried in the hearts of the microchips, a task accomplished by typing a command in DOS prompt or by clicking with your mouse on some tiny icon on the screen.

4 Such intelligence is fragile and short-lived. All those millions of microscopic switches are constantly flipping on and off in time to dashing surges of electricity. All it takes is an errant instruction or a stray misreading of a single chip to send this wonderfully intelligent golem into a state of catatonia or hit the off switch and what was a pulsing artificial life dies without a whimper. Then the next time you turn it on, birth begins all over again.

5 PCs are powerful creations that often seem to have a life of their own. Usually they respond to a seemingly magic incantation typed as a prompt or to wave of a mouse by performing tasks we couldn't imagine doing ourselves without some sort of preternatural help. There are the times when our PCs rebel and open the gates of chaos onto our natty ordered columns of numbers, our carefully made sentences, and our beautifully crafted graphics. Are we playing with power not entirely under our control?

6 A middle-aged woman sat down at a personal computer for the first time in her life. She placed her hands above the keyboard, ready to type – but hesitated. Turning, to the instructor, she asked warily: "It won't know what I'm thinking, will it?" Such concerns abound among people whose knowledge of computers comes from movies like 2001: A Space Odyssey (in which Hal, the computer with the sticky-sweet voice, tries to take control of the spaceship). Terms such as computer anxiety, and computer phobia have entered our language to describe such wariness. Many people try to avoid situations in which they might be forced into contact with computers. Even businesspeople

who deal with computers daily may experience a form of cyberphobia – fear of computers. As a result of their fear, some office workers who are cyberphobic suffer nausea, sweaty palms, and high blood pressure. Young people who have grown up with computers may not understand these reactions.

7 What are such people afraid of? Some may worry about the mathematical implications of the word computer. It seems to suggest that only a person with strong analytical and quantitative skills can use the machine. In fact, as we see more and more often, even very young children whose math skills have yet to form can use computers.

8 Some people are fearful of the computing environment. The movies love to portray old-fashioned, large computer systems – sanitized rooms walled by machines alive with blinking lights and spinning reels; it all looks intimidating. There is a notion that computers are temperamental gadgets and that, once a glitch gets into a computer system, it may wreak all kinds of havoc – from fouling up bank statements to launching nuclear missiles by mistake. Indeed, computer billing and banking errors are problems; however, most errors blamed on computers are the result of mistakes made by people. Computers do not put in the data they must work with, people do. Even so, correcting an error can be frustratingly slow.

9 Many people worry about computers in relation to their jobs. Some people doubt they have the skills to find jobs and keep them in a technological labor market. Many feel that keeping up with the swift pace of technological change is impossible because it requires costly and continuous training and development. A good many present-day executives whose companies have installed computer terminals in their offices also worry about typing – either they do not know how to type or they are afraid they will lose status if they use a keyboard.

10 Interestingly, there is another side to computer anxiety: the fear of being left out or left behind. If everyone around you is talking about, living with, and working around computers, how can you keep from revealing your limited understanding?

11 People are also nervous that computers might fall into the wrong hands. As examples of electronic wrongdoing, try these for size: An "error" purposefully introduced into your computerized credit report by someone who wanted to cause you trouble might do irreparable damage to your financial standing, ending any hopes you

might have for owning a home someday. An easily obtainable computerized list might carry personal information that could lead to an invasion of your privacy or at the least, a pile of junk mail. Think of all the forms you have filled out for schools, jobs, doctors, credit services, government offices, and so on. There is scarcely one fact related to you that is not on record in a computer file somewhere. Could unauthorized persons obtain this information?

12 Computer fraud and computer security are not simple issues; they are concerns that society must take seriously. Should we, as computer columnist John Dvorak advocates, let things work themselves out in the courts? Or, should legislators be encouraged to create laws for society's protection?

Exercise 2

Find in the text the English equivalents to:

деловые люди; страх перед компьютерами; испытывать тошно- ту; высокое кровяное давление; математический смысл (значение); старомодные компьютерные системы; выглядеть устрашающе; мерцающие огни; вращающиеся катушки; временные приспособления; по ошибке; обвинять компьютеры; исправлять ошибки; установить терминалы; использовать клавиатуру; потерять статус; попасть в «дурные» руки; нанести непоправимый ущерб; невостребованная почта; заполнить бланк; записать в компьютерный файл; создать законы для защиты общества.

Exercise 3

True or false?

1 People are not interested in computers, they just don't want to be left behind.

2 Computers are going to make many careers obsolete.

3 Most jobs will be lost because of computers.

4 Computers change the way jobs are performed.

5 People who refuse to have anything to do with computers may soon be regarded as people who refuse to learn to drive.

6 Computers are powerful, potentially dangerous tools with a life of their own.

7 Most of businesspeople write or commission their own programs.

8 Computers are now smaller and more powerful than ever before.

9 Computers have resulted in massive unemployment in many countries.

10 Managers with little or no computer experience should overrely on computers.

11 Computers can result in an invasion of people's privacy.

12 Today the challenge is to manage the information explosion through the use of well-designed information.

13 Data = information.

14 Computerization leads to elimination of workers' jobs (robots) and white-collar jobs (computers).

15 The bank computer thefts are carried out by computer whizzes who know the correct codes to use to access accounts in *order to* steal or manipulate money.

16 In a few seconds computer can make a mistake so great that it would take many months to equal it.

17 Computer monitoring of people leads to job stress and more frequent illnesses.

18 One person's error is another person's data.

19 To err is human; to really foul things up requires a computer.

Exercise 4

Give definitions to:

e.g. a system analyst is a person who identifies the information needed and develops a management info system with the assistance of computer programs.

A computer whiz (whizard), a hacker, a computer-literate person, a computer science student, a computer engineer, a computer programmer, a computer operator.

Exercise 5

Give synonyms to:

swift, costly, financial standing, to introduce into, to obtain, issue, to concern, tiny, magic, artificial, to turn on, accurately, anxiety,

fear, to lead to, old-fashioned command, to spin, to require.

Give antonyms to:

fraud, tiny, fragile, fearful, to frustrate, dead, intelligent, capable, short-lived, damage, to find jobs, slow, to foul up.

Exercise 6

Put the proper words into sentences

mistakes/errors, time, use/operation, improving, human, are, accuracy, so, part/role, make, involved, since, back, ever, replaced, more.

LESSON 4

Exercise 1

Read the text and analyze it

Why I won't buy a computer

1 I do not see that computers are bringing us one step nearer to anything that does matter to me: peace, economic justice, ecological health, political honesty, stability, good work.

2 What would a computer cost me? More money than I can afford and more than I wish to pay to people whom I do not admire. But the cost would not be just monetary. It is well understood that technological innovation always requires the discarding of "the old model", what would be superseded would be not only something, but somebody.

3 To make myself as plain as I can, I should give my standards for technological innovations in my work. They are as follows:

– The new tool should be cheaper than the one it replaces.

– It should be at least as small in scale as the one it replaces.

– It should work clearly and demonstrably better than the one it replaces.

– It should use less energy.

– If possible it should use some form of solar energy.

– It should be repairable by a person of ordinary intelligence, provided he has the necessary tools.

– It should be purchasable and repairable as near to home as possible.

– It should come from a small, privately owned shop or store that will take it back for maintenance and repair.

– It should not disrupt or replace anything good that already exists, and this includes family and community relationships.

Exercise 2

Answer the following questions:

1 What does the author think a computer would "cost" him?

2 Given the author's standards Tor technological innovation, what other new tools do you think he might object to?

3 How has technology changed your everyday life?

4 What new "gadgets" do you particularly like?

5 Have you learned to use a computer? Why or why not?

6 Do you fear the power of computers?

7 How will science and technology affect our lives in future?

8 List ten modern inventions:

Invention	Replacement	Advantage	Disadvantage
telephone	writing letters	less time	too slow
silicon chip			
mobile phone			
electricity			

Exercise 3

True or false?

Modern technology is out of control, and ruining the quality of life on Earth; we must limit technology and its influence on individual.

Modern inventions are labor-saving devices. Without them people remain slaves to boring, repetitive work.

Exercise 4

Complete the following and discuss it:

1 Scientific and technological breakthroughs have brought great benefits. You only have to look around your own home to see...

2 Many illnesses can now be treated or cured, for example, ...

3 Other examples of changes are...

4 Have our lives always been improved, however? Have we become too passive? Are we too dependent on technology? How dangerous could it be?

5 Take, for example, television, computer games, the Internet...

6 New products have also made a *major* difference to our working lives.

7 Nowadays, ...

8 In the future there may be even more major breakthroughs in the fields of medicine, leisure, work...

9 We may no longer have to...

10 We will be able to...

LESSON 5

Exercise 1

Read the text and analyze it

Programming Language

A programming language is an artificial language designed to express computations that can be performed by a machine, particularly a computer. Programming languages can be used to create programs that control the behavior of a machine, to express algorithms precisely, or as a mode of human communication.

The earliest programming languages predate the invention of the computer, and were used to direct the behavior of machines such as Jacquard looms and player pianos. Thousands of different programming languages have been created, mainly in the computer field, with many more being created every year. Most programming languages describe computation in an imperative style, i.e., as a sequence of commands, although some languages, such as those that support functional programming or logic programming, use alternative forms of description.

A programming language is usually split into the two components of syntax (form) and semantics (meaning) and many programming languages have some kind of written specification of their syntax and/or semantics. Some languages are defined by a specification document, for example, the C programming language is specified by an ISO Standard, while other languages, such as Perl, have a dominant implementation that is used as a reference.

Definitions

A programming language is a notation for writing programs, which are specifications of a computation or algorithm. Some, but not all, authors restrict the term "programming language" to those languages that can express all possible algorithms. Traits often considered important for what constitutes a programming language include:

Function and target: A computer programming language is a language used to write computer programs, which involve a computer performing some kind of computation or algorithm and possibly control external devices such as printers, disk drives, robots, and so on. For example PostScript programs are frequently created by another program to control a computer printer or display. More generally, a programming language may describe computation on some, possibly abstract, machine. It is generally accepted that a complete specification for a programming language includes a description, possibly idealized, of a machine or processor for that language. In most practical contexts, a programming language involves a computer; consequently programming languages are usually defined and studied this way. Programming languages differ from natural languages in that natural languages are only used for interaction between people, while programming languages also allow humans to communicate instructions to machines.

Abstractions: Programming languages usually contain abstractions for defining and manipulating data structures or controlling the flow of execution. The practical necessity that a programming language support adequate abstractions is expressed by the abstraction principle; this principle is sometimes formulated as recommendation to the programmer to make proper use of such abstractions.

Expressive power: The theory of computation classifies languages by the computations they are capable of expressing. All Turing complete languages can implement the same set of algorithms. ANSI/ISO SQL and Charity are examples of languages that are not Turing complete, yet often called programming languages.

Markup languages like XML or HTML, which define structured data, are not generally considered programming languages. Programming languages may, however, share the syntax with markup

languages if a computational semantics is defined. XSLT, for example, is a Turing complete XML dialect. Moreover, LaTeX, which is mostly used for structuring documents, also contains a Turing complete subset.

The term computer language is sometimes used interchangeably with programming language. However, the usage of both terms varies among authors, including the exact scope of each. One usage describes programming languages as a subset of computer languages. In this vein, languages used in computing that have a different goal than expressing computer programs are generically designated computer languages. For instance, markup languages are sometimes referred to as computer languages to emphasize that they are not meant to be used for programming. Another usage regards programming languages as theoretical constructs for programming abstract machines, and computer languages as the subset thereof that runs on physical computers, which have finite hardware resources. John C. Reynolds emphasizes that formal specification languages are just as much programming languages as are the languages intended for execution. He also argues that textual and even graphical input formats that affect the behavior of a computer are programming languages, despite the fact they are commonly not Turing-complete, and remarks that ignorance of programming language concepts is the reason for many flaws in input formats.

Exercise 2

Complete the following:

1 A programming language is an artificial language ….
2 Programming languages can be used ….
3 The earliest programming languages were used ….
4 Most programming languages describe ….
5 A programming language is usually split into ….
6 A programming language is a notation ….
7 Programming languages differ from natural languages …
8 The practical necessity is expressed by ….
9 The theory of computation classifies languages ….
10 The term computer language is sometimes used ….
11 Another usage regards ….

Exercise 3

Answer the questions:

1 What is a programming language?

2 What do programming languages describe?

3 What are the two components of programming languages?

4 What is the target of a computer programming language?

5 What may a programming language describe?

6 Programming languages usually contain abstractions, do they?

7 Are markup languages considered programming languages?

8 Is the term computer language sometimes used interchangeably with programming language?

LESSON 6

Exercise 1

Read the text and analyze it

The Virtual Reality Modeling Language

The Virtual Reality Modeling Language (VRML) is a language for describing multi-participant interactive simulations – virtual worlds networked via the global Internet and hyperlinked with the World Wide Web. All aspects of virtual world display, interaction and internetworking can be specified using VRML. It is the intention of its designers that VRML become the standard language for interactive simulation within the World Wide Web.

The first version of VRML allows for the creation of virtual worlds with limited interactive behavior. These worlds can contain objects which have hyperlinks to other worlds, HTML documents or other valid MIME types. When the user selects an object with a hyperlink, the appropriate MIME viewer is launched. When the user selects a link to a VRML document from within a correctly configured WWW browser, a VRML viewer is launched. Thus VRML viewers are the perfect companion applications to standard WWW browsers for navigating and visualizing the Web. Future versions of VRML will allow for richer behaviors, including animations, motion physics and real-time multi-user interaction.

VRML Mission Statement

The history of the development of the Internet has had three

distinct phases; first, the development of the TCP/IP infrastructure which allowed documents and data to be stored in a proximally independent way; that is, Internet provided a layer of abstraction between data sets and the hosts which manipulated them. While this abstraction was useful, it was also confusing; without any clear sense of "what went where", access to Internet was restricted to the class of sysops/net surfers who could maintain internal cognitive maps of the data space.

Next, Tim Berners-Lee's work at CERN, where he developed the hypermedia system known as World Wide Web, added another layer of abstraction to the existing structure. This abstraction provided an "addressing" scheme, a unique identifier (the Universal Resource Locator), which could tell anyone "where to go and how to get there" for any piece of data within the Web. While useful, it lacked dimensionality; there's no there within the web, and the only type of navigation permissible (other than surfing) is by direct reference. In other words, I can only tell you how to get to the VRML Forum home page by saying, "http://www.wired.com/", which is not human-centered data. In fact, I need to make an effort to remember it at all. So, while the World Wide Web provides a retrieval mechanism to complement the existing storage mechanism, it leaves a lot to be desired, particularly for human beings.

Finally, we move to "perceptualized" Internetworks, where the data has been sensualized, that is, rendered sensually. If something is represented sensually, it is possible to make sense of it. VRML is an attempt (how successful, only time and effort will tell) to place humans at the center of the Internet, ordering its universe to our whims. In order to do that, the most important single element is a standard that defines the particularities of perception. Virtual Reality Modeling Language is that standard, designed to be a universal description language for multi-participant simulations.

These three phases, storage, retrieval, and perceptualization are analogous to the human process of consciousness, as expressed in terms of semantics and cognitive science. Events occur and are recorded (memory); inferences are drawn from memory (associations), and from sets of related events, maps of the universe are created (cognitive perception). What is important to remember is that the map is not the territory, and we should avoid becoming

trapped in any single representation or world-view. Although we need to design to avoid disorientation, we should always push the envelope in the kinds of experience we can bring into manifestation!

History

VRML was conceived in the spring of 1994 at the first annual World Wide Web Conference in Geneva, Switzerland. Tim Berners-Lee and Dave Raggett organized a Birds-of-a-Feather (BOF) session to discuss Virtual Reality interfaces to the World Wide Web. Several BOF attendees described projects already underway to build three dimensional graphical visualization tools which interoperate with the Web. Attendees agreed on the need for these tools to have a common language for specifying 3D scene description and WWW hyperlinks – an analog of HTML for virtual reality. The term Virtual Reality Markup Language (VRML) was coined, and the group resolved to begin specification work after the conference. The word 'Markup' was later changed to 'Modeling' to reflect the graphical nature *of* VRML.

Shortly after the Geneva BOF session, the www-vrml mailing list was created to discuss the development of a specification for the first version of VRML. The response to the list invitation was overwhelming: within a week, there were over a thousand members. After an initial settling-in period, list moderator Mark Pesce of Labyrinth Group announced his intention to have a draft version of the specification ready by the WWW Fall 1994 conference, a mere five months away. There was general agreement on the list that, while this schedule was aggressive, it was achievable provided that the requirements for the first version were not too ambitious and that VRML could be adapted from an existing solution. The list quickly agreed upon a set of requirements for the first version, and began a search for technologies which could be adapted to fit the needs of VRML.

The search for existing technologies turned up a several worthwhile candidates. After much deliberation the list came to a consensus: the Open Inventor ASCII File Format from Silicon Graphics, Inc. The Inventor File Format supports complete descriptions of 3D scenes with polygonally rendered objects, lighting, materials, ambient properties and realism effects. A subset of the Inventor File Format, with extensions to support networking, forms the basis of VRML. Gavin Bell of Silicon Graphics has adapted the

Inventor File Format for VRML, with design input from the mailing list. SGI has publicly stated that the file format is available for use in the open market, and has contributed a file format parser into the public domain to bootstrap VRML viewer development.

Exercise 2

Discuss the problems.

1 The Virtual Reality Modeling Language (VRML) is a language for describing multi-participant interactive simulations.

2 VRML Mission Statement.

3 The history of the development.

Use the following expressions:

First of all, I'd like to tell you …; According to the text …; To tell the truth, …; As for me, …; I must admit, …:

LESSON 7

Exercise 1

Read the text and analyze it

What is a mailing list?

A mailing list is a basic type of discussion group that uses e-mail to communicate. The messages are distributed to all the subscribers, i.e. everyone that belongs to the list. There are thousands of lists covering every imaginable topic, from hobbies and music to news and science.

Types of lists

Discussion lists let you send and receive messages, providing a discussion forum for the participants; but they're different from newsgroups. In a discussion list you receive the messages directly into your e-mail box. In a newsgroup you read the articles, i.e., messages posted by contributors that are stored in one central location.

One-way lists only let you receive messages, not send them. They're good for busy people who only wish to receive broadcast information.

Some lists can be received as a periodic "digest". This contains a lot of messages which have been grouped together and sent as a single message.

Other lists are restricted to certain users, requiring specific

qualifications to join them. For example, a list about the science of stars and planets may be restricted to astronomers.

How to subscribe

Before you can start receiving messages from a list you need to subscribe to it. This process adds your e-mail address to the list.

If the list is automated (controlled by a computer program like listserv, listproc, or majordomo) you usually write in the body of the message:

<subscribe listname>

where "listname" is the name of the list. The address has the form

<majordomo@address.site>

You can quit a mailing list at any time. To unsubscribe you just need to write

<unsubscribe listname>

or, if this doesn't work, try

<signoff listname>

If the list is administered by a person, just send a message to the human moderator saying something like <please subscribe me to the list>. The address has the form

<list-request@address.site>.

Remember that each list has two addresses: (i) the administrative address, used to subscribe and unsubscribe , and (ii) the list address, used to distribute the messages to everyone on the list.

Tips

– When you join a list, you receive a confirmation message and some instructions about how to unsubscribe. Save this information, you may need it in the future.

– Some lists have a FAQ (frequently asked questions) document with the most common questions asked by newbies (new contributors). Read this file if you have any problem.

– Avoid flame wars. Flames are insulting messages directed at each other in discussion forums.

– When you go on holiday, remember to unsubscribe from lists temporarily. Otherwise your mailbox may overflow with messages.

– The Publicly Accessible Mailing Lists Website will help you find the mailing list that interests you. Go to

http://www.neosoft.com/internet/paml and choose an index by name or by subject.

– You can also try Liszt, a famous list directory, at http://www.liszt.com.

Exercise 2

How much do you know about mailing lists? Try to fill in the missing words. Then read the text more carefully.

1 A … is a discussion forum where participants subscribe to a list and receive messages via e-mail.

2 In a discussion list you receive the messages directly into your ….

In a newsgroup, however, you read the articles that are stored in one ….

3 To receive messages from a mailing list, first you need to … to it.

4 Some lists may be … to certain professionals, requiring specific qualifications to join them.

5 Each list has two addresses: (i) …, and (ii) ….

6 Mailing lists usually have a FAQ file. FAQ is an acronym for ….

Exercise 3

Complete the sentences using the text above to help you. Then write the words in the puzzle to discover the missing word.

1 A… mailing lists are controlled by a special computer program.

2 The process of subscribing adds your e… address to the list.

3 A d… is a type of list that groups individual messages together and sends them periodically as one message.

4 In newsgroups, the messages posted by contributors are called "a…".

5 Insulting or insensitive messages directed at each other in a discussion forum are known as f….

6 New participants in discussion forums are also called "n…".

7 If you want to get off a mailing list, you just type the command u… followed by the name of the list.

8 The lists a … by a human moderator usually have the word "-request" in the address.

LESSON 8

Words to be Learned

Malicious code, malicious program, malware, rogue program, virus, executable file, propagate, boot sector, worm, victim, clogging disk drives, blended threat, Trojan Horse, deceptively labeled program, logic bomb, distinguish, time bomb, hoax, attachment

Exercise 1

Read the text and analyze it

Malicious Computer Programs

Malicious code, malicious program, malware, rogue program are general terms for any computer program that is designed to harm its victim. Malicious computer programs are divided into the following classes:

– A virus is a program that *"infects" an executable file*. After infection, the executable file functions in a different way than before. There are two key features of a computer virus:

- the ability to propagate by attaching itself to executable files (e.g., application programs, operating system, macros, scripts, boot sector of a hard disk or floppy disk, etc.).

- the virus causes harm only *after* it has infected an executable file and the executable file is run.

The word "virus" is also commonly used broadly to include computer viruses, worms, and Trojan Horse programs.

– A worm is a program that *copies itself*. The distinction between a virus and worm is that a virus never copies itself – a virus is copied only when the infected executable file is run.

In the pure form a worm neither deleted nor changed files on the victim's computer. It simply made multiple copies of itself and sent those copies from the victim's computer, clogging disk drives and the Internet. Releasing such a worm into the Internet will slow the legitimate traffic on the Internet, as continuously increasing amounts of traffic are mere copies of the worm.

From early 2002, a worm could drop a virus into the victim's computer. This kind of worm became known as a *blended threat,*

because it combined two different types of malicious code.

– A Trojan Horse is a *deceptively labeled* program that contains at least one function that is unknown to the user and that harms the user. A Trojan Horse does *not* replicate, which distinguishes it from viruses and worms. Some of the more serious Trojan horses allow a hacker to remotely control the victim's computer, perhaps to collect passwords and credit card numbers and send them to the hacker, or perhaps to launch “denial of service” attacks on websites.

– A logic bomb is a program that "detonates" when some event occurs. The detonated program might stop working (e.g., go into an infinite loop), crash the computer, release a virus, delete data files, or any of many other harmful possibilities. A time bomb is a type of logic bomb, in which the program detonates when the computer's clock reaches some target date.

– A hoax is a warning about a nonexistent malicious program.

The first computer virus found "in the wild" was written in 1986 in a computer store in Lahore, Pakistan. In the 1980s, computer viruses were generally spread by passing floppy disks from one user to another. In the late 1990s, computer viruses were generally spread via the Internet, either in e-mail (e.g., a virus contained in a Microsoft Word macro, or a worm contained in an attachment to e-mail) or in programs downloaded from a website.

Exercise 2

Match the following:

1	Malicious program, malware, rogue program	A	is a program that copies itself
2	A virus	B	is a program that "detonates" when some event occurs
3	A worm	C	is a type of logic bomb, in which the program detonates when the computer's clock reaches some target date
4	A Trojan Horse	D	is a warning about a nonexistent malicious program
5	A logic bomb	E	is a program that "infects" an executable file

6	A time bomb	F	is a deceptively labeled program that contains at least one function that is unknown to the user and that harms the user
7	A hoax	G	are general terms for any computer program that is designed to harm its victim

LESSON 9

Exercise 1

Read the text and analyze it

Computer crime

Hackers: computer outlaws. People attacking computer systems are named as hackers in mass media (and accordingly in a society). However many representatives of a computer underground think that according to the history, the word "hacker" concerns the person who increases functionalities of computers. Hence, hackers are the "good" people acting with noble aims: they train a computer in performance of new functions. The use of a word "hacker" in describing the computer vandals or thieves deforms not only sense of the term, but also the historical concept of "hacking".

People, who use the term "hacker" with a positive meaning, describe the person attacking computer systems as "cracker". So, on a professional slang the hackers are "good", but crackers are "bad". However, as mass media name both their as "hackers", the term "cracker" is used very seldom.

Sometimes it is possible to see such combinations as black hat or white hat in relation to the different kinds of attacks. As in Westerns, where the "bad guys" wore black hats, and "good guys" wore white hats, the term "black hat" is used for designation of the attacking malefactor, but "white hat" is used for the expert in the field of computer safety, who tries to protect systems against breaking. "Black hat" tries to penetrate into system; a "white hat" finds weak spots and corrects defects. It is obvious, that people which work on two fronts (sometimes they attack systems, and sometimes protect them), are called "gray hats". Confusion in terminology has necessitated naming everyone who attacks the computers as "attackers". "Attackers" can

act as hackers, crackers, "white hat", "black hat", "gray hat", researcher of computer safety system and so on, all of them attack computers irrespective of their motives and aims. This problem has very important legal aspect. From the legal point of view, the court only can define "hackers", "crackers", "freackers", "carders" and etc. as the criminals.

Computer crimes: how common?
How often are they reported?

The FBI's national Computer Crimes Squad estimates that between 85 and 97 percent of computer intrusions are not even detected. In a recent test sponsored by the Department of Defense, the statistics were startling. Attempts were made to attack a total of 8932 systems participating in the test. 7860 of those systems were successfully penetrated. The management of only 390 of those 7860 systems detected the attacks, and only 19 of the managers reported the attacks.

The reason so few attacks were reported was "mainly because organizations frequently fear their employees, clients, and stockholders will lose faith in them if they admit that their computers have been attacked." Besides, of the computer crimes that *are* reported, few are ever solved.

So, are hackers a big cause of computer disasters?

According to the Computer Security Institute, these are the types of computer crime and other losses:

– Human errors – 55%

– Physical security problems – 20% (e.g., natural disasters, power problems)

– Insider attacks conducted for the purpose of profiting from computer
crime – 10%

– Disgruntled employees seeking revenge – 9%

– Viruses – 4%

– Outsider attacks – 1-3%

So when you consider that many of the outsider attacks come from professional computer criminals many of whom are employees of the competitors of the victims, hackers are responsible for almost no damage at all to computers.

In fact, on the average, it has been our experience that hackers do far more good than harm.

Yes, we are saying that the recreational hacker who just likes to play around with other people's computers is not the guy to be afraid of. It's far more likely to be some guy in a suit who is an employee of his victim.

What is a remote attack?

A *remote attack* is any attack that is initiated against a machine that the attacker does not currently have control over; that is, it is an attack against any machine other than the attacker's own (whether that machine is on the attacker's subnet or 10,000 miles away). The best way to define a remote machine is this:

A *remote machine* is any machine – other than the one you are now on-that can be reached through some protocol over the Internet or any other network or medium.

The operating system

Today, networks may harbour dozens of different machines with disparate operating systems and architecture. One would think that for the cracker, this would be a hostile and difficult-to-manage environment. Not so.

The more diverse your network nodes are (in terms of operating system and architecture), the more likely it is that a security hole exists. Each operating system has its own set of bugs. Some of these bugs are known, and some may be discovered over time. In a relatively large network, where there may be many different types of machines and software, you have a better chance of finding a hole. The system administrator is, at day's end, only a human being. He cannot be constantly reviewing security advisories for each platform in turn. There is a strong chance that his security knowledge of this or that system is weak.

For example, holes emerge each day for a wide range of platforms. True, both tools are extensible, and one can therefore add new scan modules, which eliminate this hole.

Exercise 2

Answer the questions

1 What kind of people is named as hackers?
2 Are hackers the "good" people acting with noble aims?
3 What is the difference between "hackers" and "crackers"?

4 What does "black hat" try to do?
5 What is the aim of "white hats" and "gray hats"?
6 Are hackers a big cause of computer disasters?
7 What is your attitude to the problem?

LESSON 10

Read the list of international words

Object, concept, code, message, communicate, technology, characteristics, function, method, animation, control, electronic, action, indicate, diagram, encapsulation, idea, design, system, component, interaction, functionality, parameter.

Exercise 1

Read the text and analyze it

Object-Oriented Programming Concepts

If you've never used an object-oriented language before, you need to understand the underlying concepts before you begin writing code. You need to understand what an object is, what a class is, how objects and classes are related, and how objects communicate by using messages.

What Is an Object?

Objects are keys to understanding *object-oriented* technology. You can look around you now and see many examples of real-world objects: your dog, your desk, your television set, your bicycle.

These real-world objects share two characteristics: They all have *state* and *behavior*. For example, dogs have state (name, color and breed, hungry) and behavior (barking, fetching, and wagging tail). Bicycles have state (current gear, current pedal cadence, two wheels and number of gears) and behavior (braking, accelerating, slowing down and changing gears).

Software objects are modeled after real-world objects in that they too have state and behavior. A software object maintains its state in one or more *variables*. A variable is an item of data named by an identifier. A software object implements its behavior with *methods*. A method is a function (subroutine) associated with an object.

Definition: An object is a software bundle of variables and related methods.

You can represent real-world objects by using software objects. You might want to represent real-world dogs as software objects in an animation program or a real-world bicycle as software object in the program that controls an electronic exercise bike. You can also use software objects to model abstract concepts. For example, an *event* is a common object used in GUI window systems to represent the action of a user pressing a mouse button or a key on the keyboard.

Everything that the software object knows (state) and can do (behavior) is expressed by the variables and the methods within that object. A software object that modeled your real-world bicycle would have variables that indicated the bicycle's current state: its speed is 10 mph, its pedal cadence is 90 rpm, and its current gear is the 5th gear. These variables are formally known as *instance variables* because they contain the state for a particular bicycle object, and in object-oriented terminology, a particular object is called an *instance*.

In addition to its variables, the software bicycle would also have methods to brake, change the pedal cadence, and change gears. (The bike would not have a method for changing the speed of the bicycle, as the bike's speed is just a side effect of what gear it's in, how fast the rider is pedaling, whether the brakes are on, and how steep the hill is.) These methods are formally known as *instance methods* because they inspect or change the state of a particular bicycle instance.

The object diagrams show that the object's variables make up the center, or nucleus, of the object. Methods surround and hide the object's nucleus from other objects in the program. Packaging an object's variables within the protective custody of its methods is called *encapsulation*. This conceptual picture of an object-a nucleus of variables packaged within a protective membrane of methods-is an ideal representation of an object and is the ideal that designers of object-oriented systems strive for. However, it's not the whole story. Often, for practical reasons, an object may wish to expose some of its variables or hide some of its methods. In the Java programming language, an object can specify one of four access levels for each of its variables and methods. The access level determines which other objects and classes can access that variable or method. Variable and method access in Java is covered in Controlling *Access to Members of a Class*. Encapsulating related variables and methods into a neat

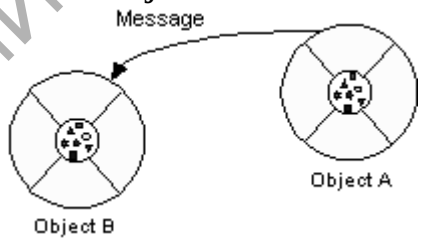
software bundle is a simple yet powerful idea that provides two primary benefits to software developers:

- *Modularity:* The source code for an object can be written and maintained independently of the source code for other objects. Also, an object can be easily passed around in the system. You can give your bicycle to someone else, and it will still work.
- *Information hiding:* An object has a public interface that other objects can use to communicate with it. The object can maintain private information and methods that can be changed at any time without affecting the other objects that depend on it. You don't need to understand the gear mechanism on your bike to use it.

What Is a Message?

A single object alone is generally not very useful. Instead, an object usually appears as a component of a larger program or application that contains many other objects. Through the interaction of these objects, programmers achieve higher-order functionality and more complex behavior. Your bicycle hanging from a hook in the garage is just a bunch of titanium alloy and rubber; by itself, the bicycle is incapable of any activity. The bicycle is useful only when another object (you) interacts with it (pedal).

Software objects interact and communicate with each other by sending *messages* to each other. When object A wants object B to perform one of B's methods, object A sends a message to object B



Sometimes, the receiving object needs more information so that it knows exactly what to do; for example, when you want to change gears on your bicycle, you have to indicate which gear you want. This information is passed along with the message as *parameters*.

These are the three components that comprise a message:

1 The object to which the message is addressed (YourBicycle)
2 The name of the method to perform (changeGears)
3 Any parameters needed by the method (lowerGear)

These three components are enough information for the receiving object to perform the desired method. No other information

or context is required.

Messages provide two important benefits.

– An object's behavior is expressed through its methods, so (aside from direct variable access) message passing supports all possible interactions between objects.

Objects don't need to be in the same process or even on the same machine to send and receive messages back and forth to each other.

Exercise 2

Agree or disagree

Objects are keys to understanding *object-oriented* technology.

These real-world objects share ten characteristics.

Software objects are modeled after imaginary objects.

An object is a software bundle of variables and related methods.

Everything that the software object knows (state) and can do (behavior) is expressed by equations.

Software objects interact and communicate with each other by sending *messages* to each other.

There are the three components that comprise a message.

Messages provide two important benefits.

Exercise 3

Answer the questions

1 What is an object?
2 How can you characterize *state* and *behavior* of the object?
3 How is software object modeled?
4 What is a method?
5 How can you represent real-world objects?
6 What is a message?
7 What are the components that comprise a message?
8 What important benefits do messages provide?

Exercise 4

Make a selective retelling of the text.

Exercise 5

Define various meanings of the following words and word combinations

Current gear, current cadence, two wheels, behavior, change gears, brake, manufacturer, advantage, rectangles, employee records, allocate, invoke, blueprint, inheritance, entity, enforce, tracking number, retail.

Exercise 6

Spot the international words in the text and guess their meaning

Exercise 7

Read the text and analyze it

What Is a Class?

In the real world, you often have many objects of the same kind. For example, your bicycle is just one of many bicycles in the world. Using object-oriented terminology, we say that your bicycle object is an *instance* of the class of objects known as bicycles. Bicycles have some state (current gear, current cadence, two wheels) and behavior (change gears, brake) in common. However, each bicycle's state is independent of and can be different from that of other bicycles.

When building bicycles, manufacturers take advantage of the fact that bicycles share characteristics, building many bicycles from the same blueprint. It would be very inefficient to produce a new blueprint for every individual bicycle manufactured.

In object-oriented software, it's also possible to have many objects of the same kind that share characteristics: rectangles, employee records, video clips, and so on. Like the bicycle manufacturers, you can take advantage of the fact that objects of the same kind are similar and you can create a blueprint for those objects. A software blueprint for objects is called a *class*.

Definition: A class is a blueprint, or prototype, that defines the variables and the methods common to all objects of a certain kind.

The class for our bicycle example would declare the instance variables necessary to contain the current gear, the current cadence, and so on, for each bicycle object. The class would also declare and provide implementations for the instance methods that allow the rider to change gears, brake, and change the pedaling cadence.

After you've created the bicycle class, you can create any number of bicycle objects from the class. When you create an instance of a class, the system allocates enough memory for the object and all

its instance variables. Each instance gets its own copy of all the instance variables defined in the class.

In addition to instance variables, classes can define *class variables*. A class variable contains information that is shared by all instances of the class. For example, suppose that all bicycles had the same number of gears. In this case, defining an instance variable to hold the number of gears is inefficient; each instance would have its own copy of the variable, but the value would be the same for every instance. In such situations, you can define a class variable that contains the number of gears. All instances share this variable. If one object changes the variable, it changes for all other objects of that type. A class can also declare *class methods*. You can invoke a class method directly from the class, whereas you must invoke instance methods on a particular instance.

Understanding Instance and Class Members discusses instance variables and methods and class variables and methods in detail.

Objects vs. Classes

You probably noticed that the illustrations of objects and classes look very similar. And indeed, the difference between classes and objects is often the source of some confusion. In the real world, it's obvious that classes are not themselves the objects they describe: A blueprint of a bicycle is not a bicycle. However, it's a little more difficult to differentiate classes and objects in software. This is partially because software objects are merely electronic models of real-world objects or abstract concepts in the first place. But it's also because the term "object" is sometimes used to refer to both classes and instances.

What Is Inheritance?

Generally speaking, objects are defined in terms of classes. You know a lot about an object by knowing its class. Even if you don't know what a penny-farthing is, if I told you it was a bicycle, you would know that it had two wheels, handle bars, and pedals.

Object-oriented systems take this a step further and allow classes to be defined in terms of other classes. For example, mountain bikes, racing bikes, and tandems are all kinds of bicycles. In object-oriented terminology, mountain bikes, racing bikes, and tandems are all *subclasses* of the bicycle class. Similarly, the bicycle class is the *superclass* of mountain bikes, racing bikes, and tandems.

Each subclass *inherits* state (in the form of variable declarations) from the superclass. Mountain bikes, racing bikes, and tandems share some states: cadence, speed, and the like. Also, each subclass inherits methods from the superclass. Mountain bikes, racing bikes, and tandems share some behaviors: braking and changing pedaling speed, for example.

However, subclasses are not limited to the state and behaviors provided to them by their superclass. Subclasses can add variables and methods to the ones they inherit from the superclass. Tandem bicycles have two seats and two sets of handle bars; some mountain bikes have an extra set of gears with a lower gear ratio.

Subclasses can also override inherited methods and provide specialized implementations for those methods. For example, if you had a mountain bike with an extra set of gears, you would override the "change gears" method so that the rider could use those new gears.

You are not limited to just one layer of inheritance. The inheritance tree, or class hierarchy, can be as deep as needed. Methods and variables are inherited down through the levels. In general, the farther down in the hierarchy a class appears, the more special its behavior.

The Object class is at the top of class hierarchy, and each class is its descendant (directly or indirectly). A variable of type Object can hold a reference to any object, such as an instance of a class or an array. Object provides behaviors that are required of all objects running in the Java Virtual Machine. For example, all classes inherit Object's to String method, which returns a string representation of the object.

Inheritance offers the following benefits:

– Subclasses provide specialized behaviors from the basis of common elements provided by the superclass. Through the use of inheritance, programmers can reuse the code in the superclass many times.

– Programmers can implement superclasses called abstract classes that define "generic" behaviors. The abstract superclass defines and may partially implement the behavior, but much of the class is undefined and unimplemented. Other programmers fill in the details with specialized subclasses.

What Is an Interface?

In English, an interface is a device or a system that unrelated entities use to interact. According to this definition, a remote control is an interface between you and a television set, the English language is an interface between two people, and the protocol of behavior enforced in the military is the interface between people of different ranks. Within the Java programming language, an *interface* is a device that unrelated objects use to interact with each other. An interface is probably most analogous to a protocol (an agreed on behavior). In fact, other object-oriented languages have the functionality of interfaces, but they call their interfaces protocols.

The bicycle class and its class hierarchy define what a bicycle can and cannot do in terms of its "bicycleness." But bicycles interact with the world on other terms. For example, a bicycle in a store could be managed by an inventory program. An inventory program doesn't care what class of items it manages as long as each item provides certain information, such as price and tracking number. Instead of forcing class relationships on otherwise unrelated items, the inventory program sets up a protocol of communication. This protocol comes in the form of a set of constant and method definitions contained within an interface. The inventory interface would define, but not implement, methods that set and get the retail price, assign a tracking number, and so on.

To work in the inventory program, the bicycle class must agree to this protocol by implementing the interface. When a class implements an interface, the class agrees to implement all the methods defined in the interface. Thus, the bicycle class would provide the implementations for the methods that set and get retail price, assign a tracking number, and so on.

You use an interface to define a protocol of behavior that can be implemented by any class anywhere in the class hierarchy. Interfaces are useful for the following:

- Capturing similarities among unrelated classes without artificially forcing a class relationship.
- Declaring methods that one or more classes are expected to implement.
- Revealing an object's programming interface without

revealing its class.

Exercise 8

1 Look through the text and pick out key words for further retelling.

2 Pick out 10-15 sentences which convey the basic information.

3 Make a selective retelling of the text.

LESSON 11

Exercise 1

Read the following words and word combinations, translate them into Russian. Consult the dictionary. Find sentences with these words in the text.

Webpage hyperlink, browser, request, determine, route, work out.

Exercise 2

Read the text and analyze it

The World Wide Web

To find the webpage you want, you have to click on a webpage hyperlink or enter a URL, a Uniform Resource Locator into a browser. The URL is the address of the page. When you do that, the browser sends the URL to a DNS server.

The DNS server is the Domain Name Server. It uses a look-up table to find the IP address of the Web server referred to in the URL. The IP address is a unique, 32-bit, set of numbers. Every computer on the Web has its own IP address.

Once the DNS server has found the IP address, it sends it back to the browser.

The browser then uses this IP address to send a request to the Web server. The request is sent as a series of separate data packets which include both the IP address of the Web server and the IP address of the browser computer. These data packets are first sent to a router computer, which uses the IP address of the Web server to determine the best available route for each packet.

The packets are passed from router to router until they reach the Web server. They may travel by different routes before reaching the server.

As the individual packets reach the Web server, they're put back together again.

The Web server now services the request by sending the requested webpage back to the browser computer. Again it travels as a series of separate data packets from router to router. This time the router uses the IP address of the browser computer to work out the best available path for each packet. As the packets arrive at the browser computer they're combined to form the webpage you requested and are displayed in your browser.

Exercise 3

Read the title of the text and try to figure out its contents.

Exercise 4

Make a commentary of that part of the text which concerns a browser.

Find the sentences which convey the basic information. Retell the text.

LESSON 12

Words to be Learned

Linux kernel, install, the most prominent examples, source, collaboration, modify, redistribute, distribution, derivative, proprietary.

Exercise 1

Read the text and analyze it

Linux

Linux refers to the family of Unix-like computer operating systems using the Linux kernel. Linux can be installed on a wide variety of computer hardware, ranging from mobile phones, tablet computers and video game consoles, to mainframes and supercomputers. Linux is a leading server operating system.

The development of Linux is one of the most prominent examples of free and open source software collaboration; typically all the underlying source code can be used, freely modified, and redistributed, both commercially and non-commercially, by anyone under licenses such as the GNU General Public License. Typically

Linux is packaged in a format known as a Linux distribution for desktop and server use. Some popular mainstream Linux distributions include Debian (and its derivatives such as Ubuntu), Fedora and openSUSE. Linux distributions include the Linux kernel and supporting utilities and libraries to fulfil the distribution's intended use.

The name "Linux" comes from the Linux kernel, originally written in 1991 by Linus Torvalds. The main supporting user space system tools and libraries from the GNU Project (announced in 1983 by Richard Stallman) are the basis for the Free Software Foundation's preferred name GNU/Linux.

Currently most distributions include a graphical user environment, with the two most popular environments being GNOME (which can utilize additional shells such as the default GNOME Shell and UbuntuUnity) and the KDE Plasma Desktop. Many popular applications are available for a wide variety of operating systems. For example Mozilla Firefox, and OpenOffice.org have downloadable versions for all major operating systems. Furthermore, some applications were initially developed for Linux, such as Pidgin, and GIMP, and were ported to other operating systems including Windows and Mac OS X. In addition, a growing number of proprietary desktop applications are also supported on Linux.

Many types of applications available for Microsoft Windows and Mac OS X are also available for Linux. Commonly, either a free software application will exist which does the functions of an application found on another operating system, or that application will have a version that works on Linux, such as with Skype and some video games. Furthermore, the Wine project provides a Windows compatibility layer to run unmodified Windows applications on Linux.

Installing, updating and removing software in Linux is typically done through the use of package managers such as the Synaptic Package Manager, PackageKit, Pacman and Yum Extender. While most major Linux distributions have extensive repositories, often containing tens of thousands of packages, not all the software that can run on Linux is available from the official repositories. Alternatively, users can install packages from unofficial repositories, download pre-compiled packages directly from websites, or compile the source code

by themselves. All these methods come with different degrees of difficulty.

Exercise 2

True or False

1 Linux refers to the family of Unix-like computer operating systems using the Windows NT kernel.

2 Linux is a leading desktop operating system.

3 Many types of applications available for Microsoft Windows and Mac OS X are also available for Linux.

4 Installing, updating and removing software in Linux is typically done through the use of package managers.

5 Skype does not work on Linux.

Exercise 3

Multiple Choice

1 What Linux distributions are popular mainstream?

a) Debian

b) Ubuntu

c) Slackware

d) Fedora

2 The name "Linux" comes from the Linux kernel, originally written in 1991 by...

a) Bill Gates

b) Andrew Tanenbaum

c) Richard Stallman

d) Linus Torvalds

3 These applications were initially developed for Linux:

a) Firefox

b) Pidgin

c) uTorrent

d) GIMP

4 … allows to run unmodified Windows applications on Linux.

a) OpenOffice

b) GNOME Shell

c) Wine

d) KDE Plasma desktop

5 What package managers are used in Linux?

a) Synaptic
b) Pacman
c) Apple Store
d) Android Market

Exercise 4

Open Ended

1 When was the Linux kernel written?

2 What graphical user environment for Linux do you know?

3 Are any proprietary desktop applications supported on Linux?

4 Does all the software that can run on Linux is available from the official repositories?

5 Describe your relationship with Linux.

Учебное издание

Короткая Марианна Викторовна

Петухова Галина Николаевна

Хорсун Ирина Александровна

Чернякова Евгения Анатольевна

АНГЛИЙСКИЙ ЯЗЫК

ПРАКТИЧЕСКОЕ РУКОВОДСТВО

для студентов специальности 1-40 01 01 «Программное обеспечение информационных технологий»

В авторской редакции

Лицензия №02330/0133208 от 30.04.04.
Подписано в печать . .11. Формат 60x84 1/16.
Бумага писчая №1. Гарнитура «Таймс». Усл.п.л. , .
Уч.-изд.л. , . Тираж экз. Заказ № .

Отпечатано с оригинала-макета на ризографе
учреждения образования
«Гомельский государственный университет
имени Франциска Скорины».
Лиц №02330/0056611 от 16.04.04.
246019, г. Гомель, ул. Советская, 104.