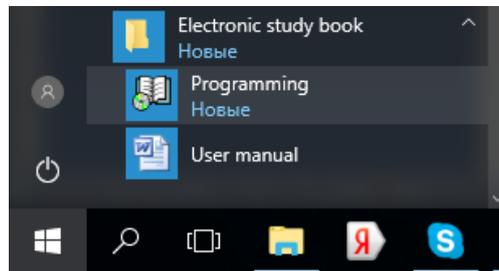
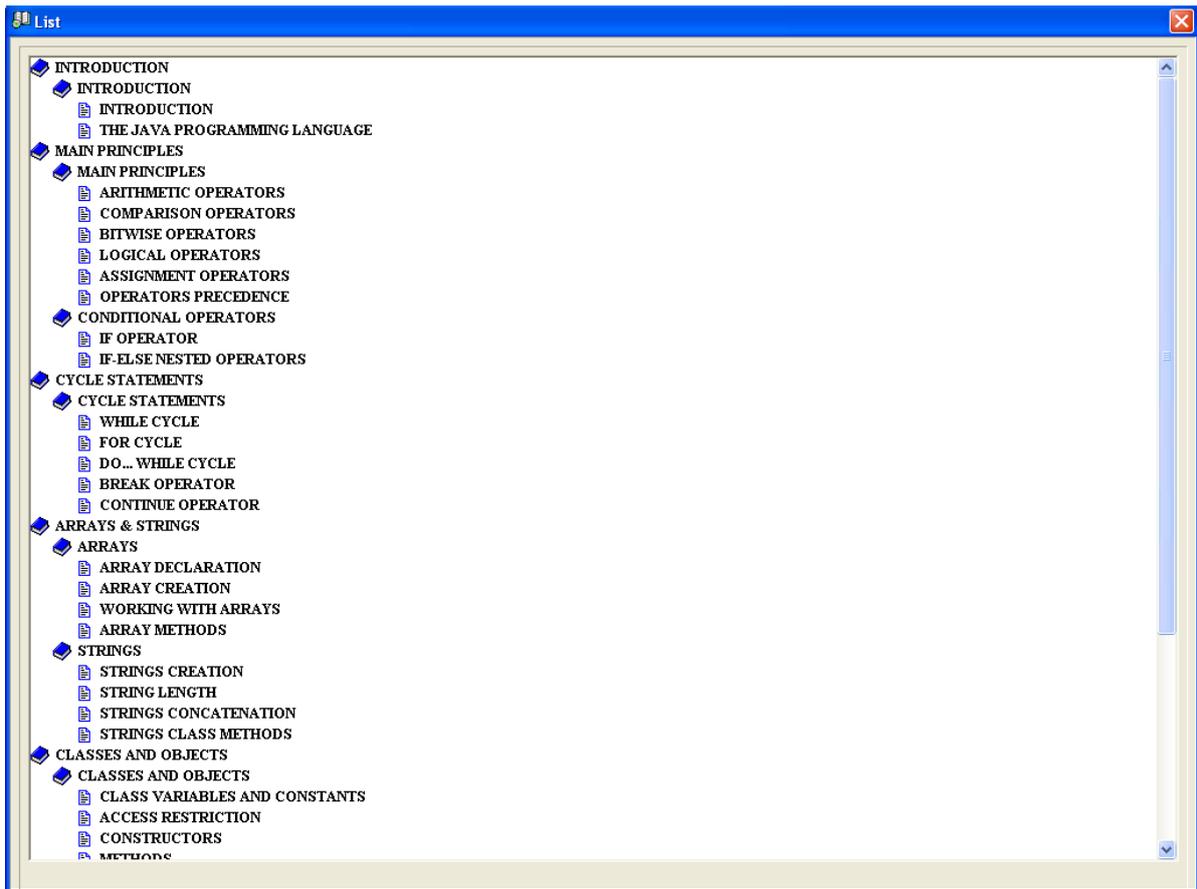


User manual of ESB (electronic study book) «Programming»

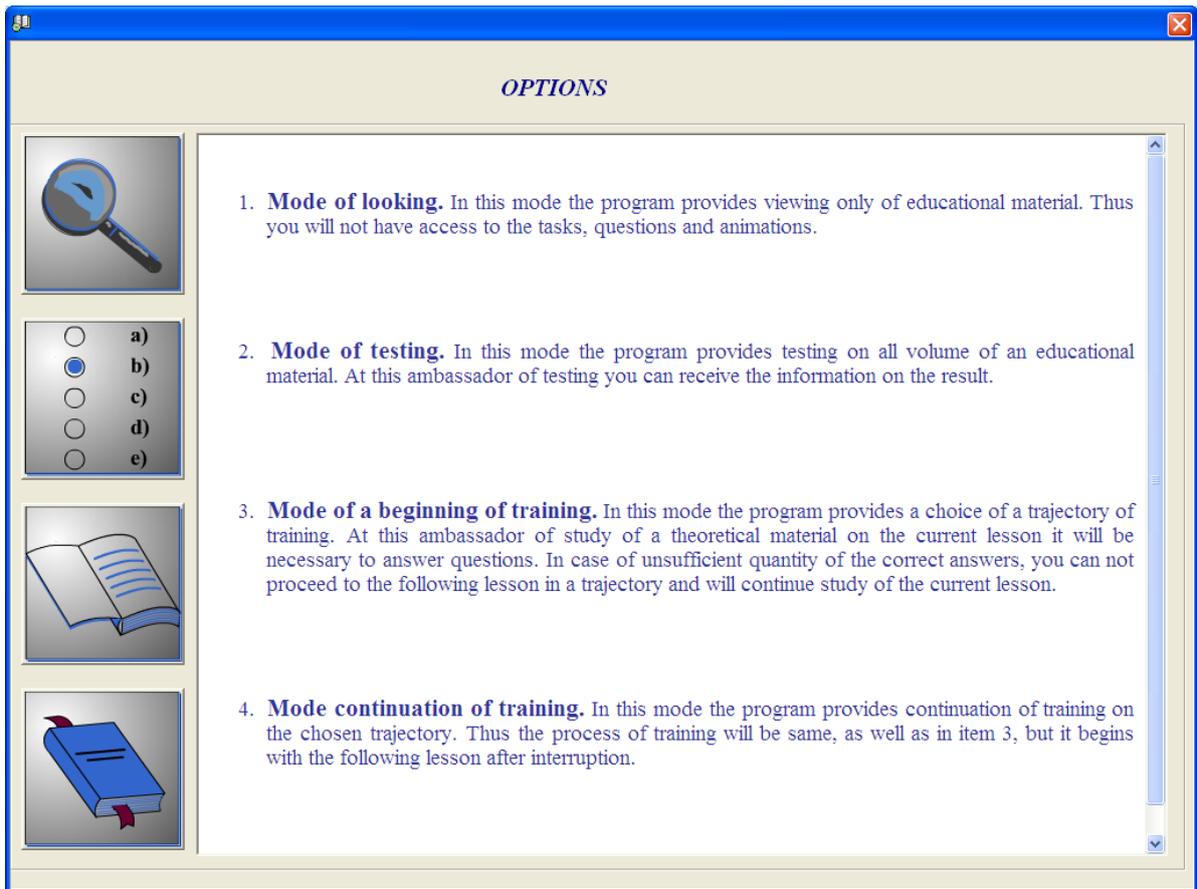
There will be a group “**Electronic study book**” where will be a shortcut “**Programming**” after installation ESB in the main menu.



It is a window which shows a structure of ESB by pressing “**Content**” button.



The trainee can choose an operating mode by pressing «Content» button.



The first is view mode. The training program provides only review of training material. There is no access to tasks, questions and files of multimedia in this mode.

Looking

- INTRODUCTION
 - INTRODUCTION
 - THE JAVA PROGRAMMING LANGUAGE
- MAIN PRINCIPLES
 - MAIN PRINCIPLES
 - ARITHMETIC OPERATORS
 - COMPARISON OPERATORS
 - BITWISE OPERATORS
 - LOGICAL OPERATORS
 - ASSIGNMENT OPERATORS
 - OPERATORS PRECEDENCE
 - CONDITIONAL OPERATORS
 - IF OPERATOR
 - IF-ELSE NESTED OPERATORS
 - CYCLE STATEMENTS
 - CYCLE STATEMENTS
 - WHILE CYCLE
 - FOR CYCLE
 - DO... WHILE CYCLE
 - BREAK OPERATOR
 - CONTINUE OPERATOR
 - ARRAYS & STRINGS
 - ARRAYS
 - ARRAY DECLARATION
 - ARRAY CREATION
 - WORKING WITH ARRAYS
 - ARRAY METHODS
 - STRINGS
 - STRINGS CREATION
 - STRING LENGTH
 - STRINGS CONCATENATION
 - STRINGS CLASS METHODS
 - CLASSES AND OBJECTS
 - CLASSES AND OBJECTS
 - CLASS VARIABLES AND CONSTANT
 - ACCESS RESTRICTION
 - CONSTRUCTORS

Arithmetical operators in Java are used in mathematical expressions in the same way as they are used in algebra. We will assume that the integer variable A is equal to 10 and the variable B is equal to 20. In the following table the arithmetical operators are listed:

Operator	Description	Example
+	Adds the values on both sides from the operator	A + B will be 30
-	Subtracts the right operand from the left one	A - B will be -10
*	Multiplies the values on both sides from operator	A * B will be 200
/	The operator of division divides the left operand into the right operand	B / A will be 2
%	Divides the left operand into the right operand and returns the excess	B % A will be 0
++	The increment increases the value of an operand by 1	B++ will be 21
--	Decrement - reduces value of an operand by 1	B-- will be 19

Example
The following simple example shows program-arithmetic operators. Copy and insert the following java-code into the test.java file. Compile and start this program:

```
public class Test {
    public static void main(String args[] ) {
        int a = 10;
        int b = 20;
        int c = 25;
        int d = 25;
        System.out.println("a + b = " + (a + b) );
        System.out.println("a - b = " + (a - b) );
        System.out.println("a * b = " + (a * b) );
        System.out.println("b / a = " + (b / a) );
        System.out.println("b % a = " + (b % a) );
        System.out.println("c % a = " + (c % a) );
        System.out.println("a++ = " + (a++) );
        System.out.println("b-- = " + (a--));
    }
}
```

The second mode is testing. The training program provides testing of all training material in this mode. Thus after testing it is possible to obtain information of testing result.

Quantity of questions: 38 Question 1

When and what company has developed the Java Language?

Sun Microsystems in 1995 year

Apple Computer in 1986 year

Generetum Software in 1999 year

Apple in 1998 year

Netscape Communication in 1989 year

Previous Next OK

The third mode is training. At first trainee must register.

Registration

Select name:

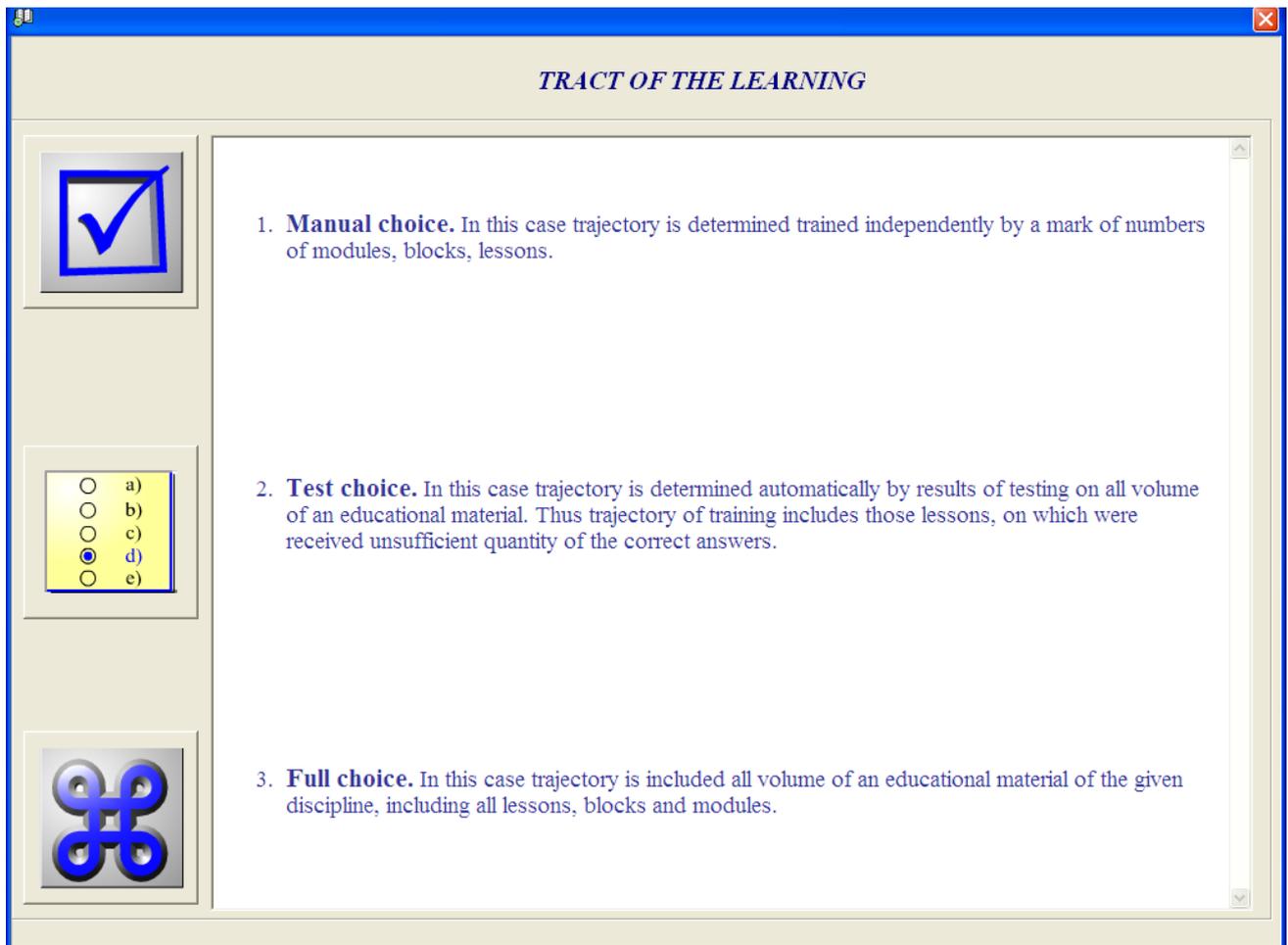
Enter name:

 Login:

OK

In this mode the training program provides the choice of a training trajectory. Thus after studying of theoretical material it will be necessary to answer test questions of the current lesson. In case of insufficient number of the correct answers the trainee won't be able to pass on to the following lesson into trajectories and will continue studying of the current lesson. It is provided an intermediate testing except the current testing (in passing to the following block), midterm testing (in passing to the following module) and final test (at training completion).

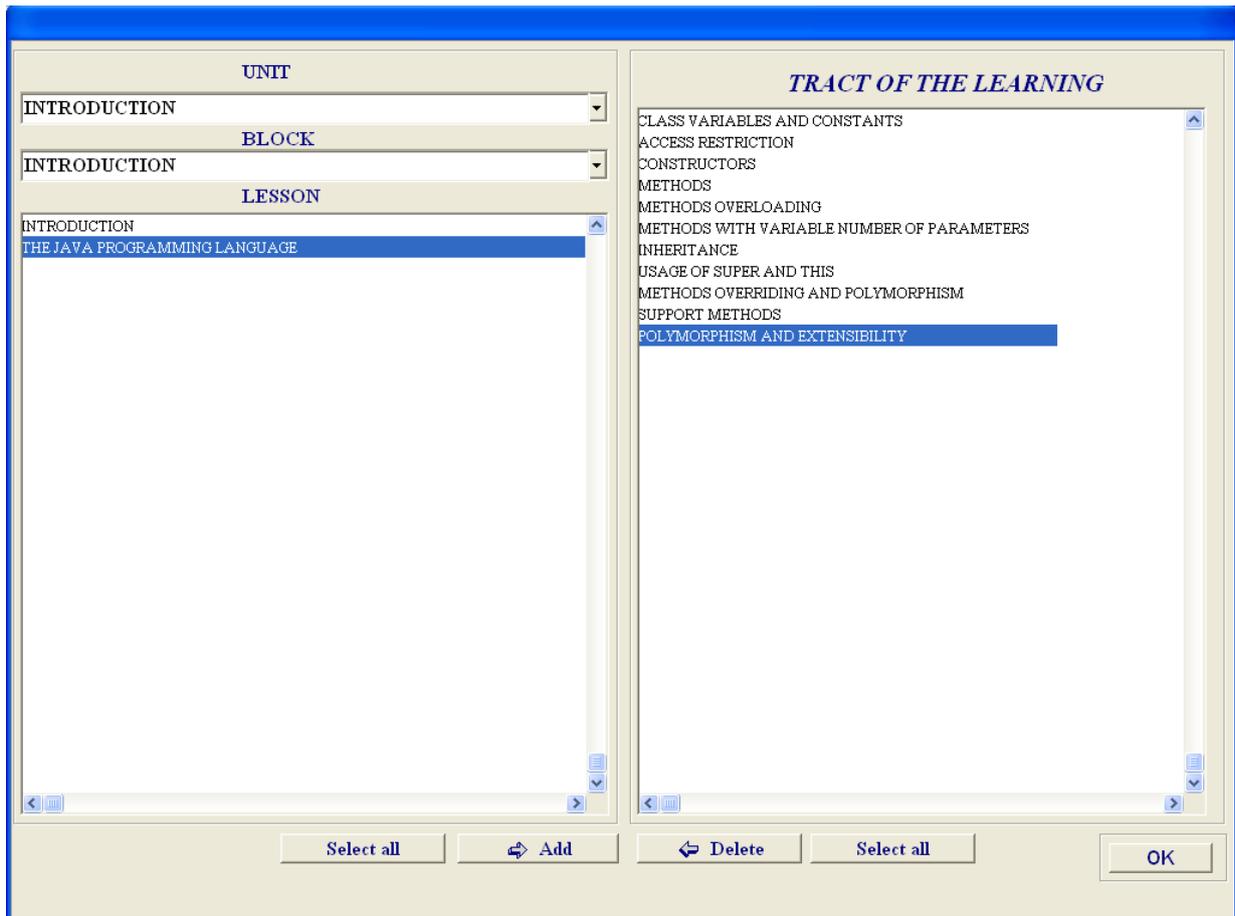
The fourth mode is training continuation. In this mode the training program provides continuation of training in the selected trajectory.



At the same time the training process begins with the following lesson after break.

The training start mode allows choosing one of the three trajectories of training: manual choice, test choice and full choice.

At the manual choice the trajectory is determined by trainee independently marking numbers of modules, blocks, lessons.



At the test choice the trajectory is determined automatically according to the results of testing of all training material. In this case the trajectory of training contains only those lessons where there are not enough correct answers.

At the full choice the trajectory includes all training material of this discipline, including all lessons, modules and blocks. After the definition of trajectory the user passes directly to a training session.

UNIT CLASSES AND OBJECTS
 BLOCK CLASSES AND OBJECTS
 LESSON CLASS VARIABLES AND CONSTANTS

Example Tasks Question Reference Thesaurus Tests

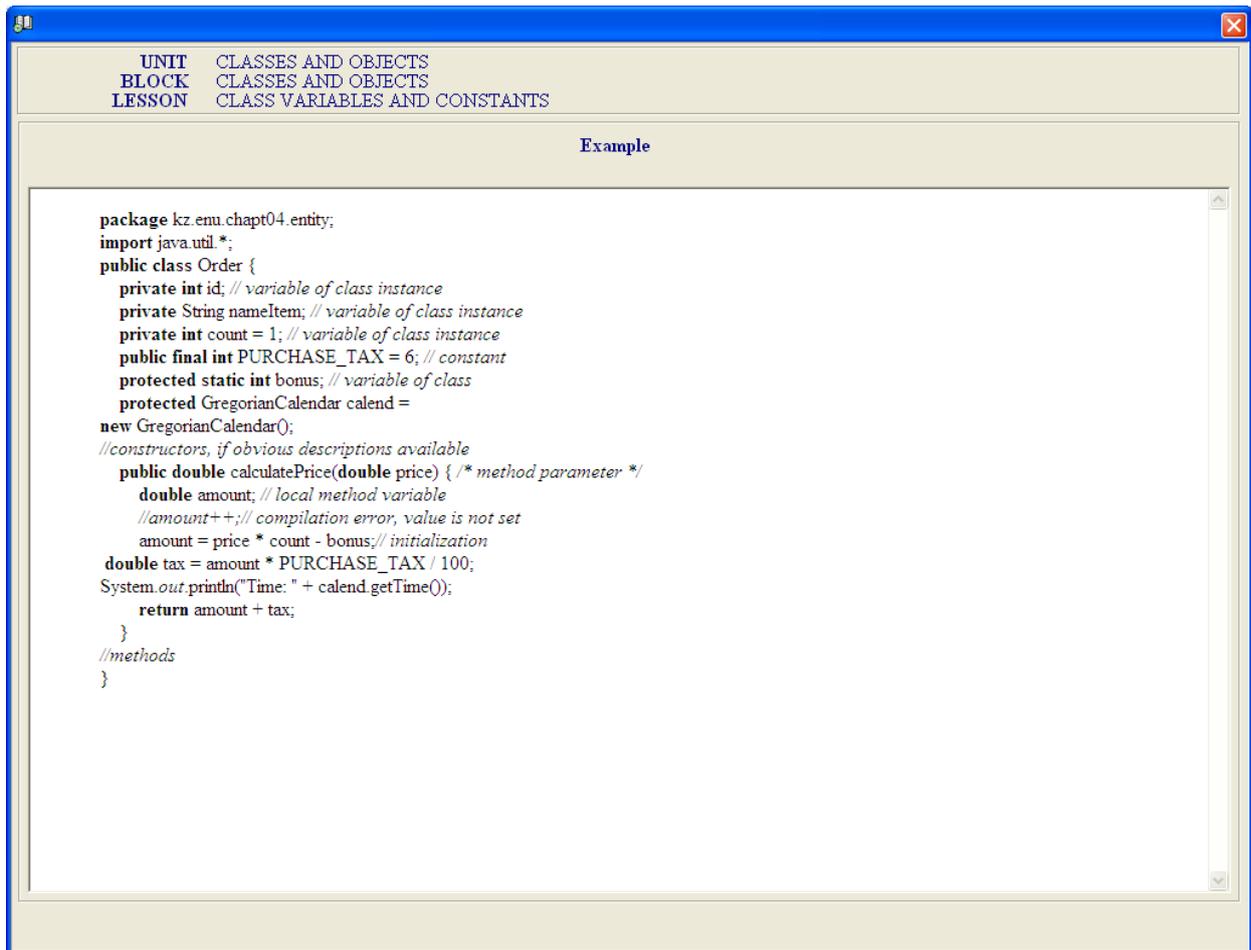
The class represents a description of set of objects with general attributes, methods, relationships and semantics.
 The classes are a basic element of abstraction of the Java language, which basic purpose, except implementing the contract assigned to it, is hiding of realization. Classes always interact with each other and are united in packages. The modules are produced from packages, which interact with each other only via a limited quantity of methods and classes without having any idea of the processes happening within other modules.
 The class name in a package should be unique. Physically the package represents a catalogue which contains the program files with classes realization.
 The classes allow implementing the decomposition of difficult system behavior to a set of elementary interactions of the connected objects. The class defines a structure and/or behavior of some element of a subject domain for which the program model is developed.
 The definition of the primitive class has the following form:
 Class **ClassName** {
 { } //logical blocks
 //constructors
 //inner classes
 // amicable data and methods
private // private data and methods
protected // protected data and methods
public // public data and methods
 }

Class variables and constants

The classes encapsulate variables and methods – members of a class. The class variables are announced in it as follows:
specifier type name;
 There can be used the static variables of a class which announced once for the whole class with a **static** specifier and identical to all copies (objects) of a class or the class sample variables being constructed for each object of a class in the Java language. The fields of a class are announced with an access specifiers **public**, **private**, **protected** or by default without specifier. Except data which are the members of a class the local variables and parameters of methods are used in class methods. Unlike the class variables encapsulated by zero elements the variables of methods are not initialized by default.
 The variables with a **final** specifier are constants. The **final** specifier can be used for the variable announced in a method and also for method parameter.

In each lesson there is a theoretical material, task for trainee self-work such as questions, thesaurus, reference book, tests.

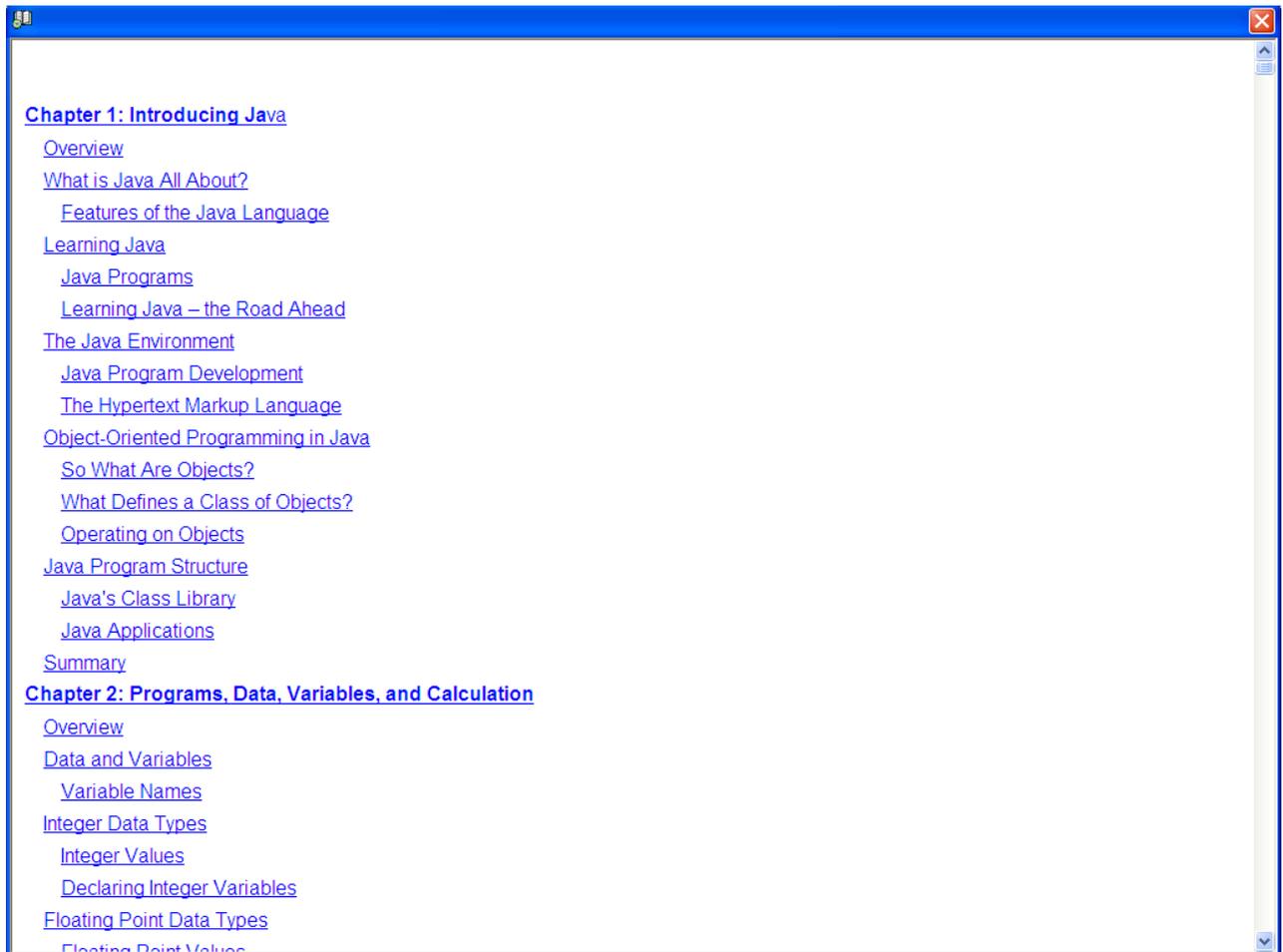
By pressing “Examples” button there will be a window with examples.



The screenshot shows a window titled "UNIT CLASSES AND OBJECTS", "BLOCK CLASSES AND OBJECTS", and "LESSON CLASS VARIABLES AND CONSTANTS". Below the title bar, the word "Example" is centered. The main area contains the following Java code:

```
package kz.emu.chapt04.entity;
import java.util.*;
public class Order {
    private int id; // variable of class instance
    private String nameItem; // variable of class instance
    private int count = 1; // variable of class instance
    public final int PURCHASE_TAX = 6; // constant
    protected static int bonus; // variable of class
    protected GregorianCalendar calend =
new GregorianCalendar();
//constructors, if obvious descriptions available
    public double calculatePrice(double price) { /* method parameter */
        double amount; // local method variable
        //amount++; // compilation error, value is not set
        amount = price * count - bonus; // initialization
        double tax = amount * PURCHASE_TAX / 100;
        System.out.println("Time: " + calend.getTime());
        return amount + tax;
    }
//methods
}
```

By pressing “Question” button there will be a window with questions.
It is necessary to press “Reference” button to review reference information.



The training element "Tests" provides an access to testing which is intended for knowledge self-checking of the current unit of training.

"Thesaurus" button provides an access to the glossary of terms and abbreviations which can be in ESB.

