Review

Introduction to Java Programming

刘艳霞
cslyx@scut.edu.cn

Content

Chapter 1
Introduction to Computers,programs and Java

1. Java is platform independent, meaning that you can write a program once and run it anywhere.
2. Java source files end with the .java extension.
3. Every class is compiled into a separate bytecode file that has the same name as the class and ends with the .class extension.
4. JVM/ JRE /JDK
5. To compile a Java source-code file from the command line, use the javac command.
6. To run a Java class from the command line, use the java command.
7. Every Java program is a set of class definitions. The keyword class introduces a class definition. The contents of the class are included in a block.
8. A Java program must have a main method. The main method is the entry point where the program starts when it is executed.
9. Java source programs are case sensitive.
10. There are two types of import statements: *specific import and wildcard import.* The specific import specifies a single class in the import statement. The wildcard import imports all the classes in a package.

Chapter 2 Elementary Programming

- A named constant is declared by using the keyword final. By convention, constants are named in uppercase.
- An identifier is a sequence of characters that consist of letters, digits, _, $. It cannot start with a digit . It cannot be a keyword.
- A literal is a constant value of boolean, numeric, character, or string data which can be assigned to the variable.
- Java provides four integer types (byte, short, int, long) that represent integers of four different sizes.
- Java provides two floating-point types (float, double) that represent floating-point numbers of two different precisions. Note: The arithmetic of floating-point data is not accurate.
- Character type (char) represents a single character.
- Java provides operators that perform numeric operations: + (addition), – (subtraction), * (multiplication), / (division), and % (remainder).Java provides shorthand operators += (addition assignment), –= (subtraction assignment), *= (multiplication assignment), /= (division assignment), and %= (remainder assignment).
- The increment operator (++) and the decrement operator (– –) increment or decrement a variable by 1.

8. When evaluating an expression with values of mixed types, Java automatically converts the operands to appropriate types.
9. Casting a variable of a type with a small range to a variable of a type with a larger range is known as *widening a type*.
10. Casting a variable of a type with a large range to a variable of a type with a small range is known as *narrowing a type*.
11. Widening a type can be performed automatically without explicit casting. Narrowing a type must be performed explicitly.
12. You can explicitly convert a value from one type to another using the (type)exp notation.

Chapter 3 Selections

1. A boolean variable stores a true or false value.
2. The relational operators (<, <=, ==, !=, >, >=) work with numbers and characters, and yield a Boolean value.
3. The Boolean operators &&, ||, !, and ^ operate with Boolean values and variables.
   - When evaluating p1 && p2, Java first evaluates p1 and then evaluates p2 if p1 is true; if p1 is false, it does not evaluate p2.
   - When evaluating p1 || p2, Java first evaluates p1 and then evaluates p2 if p1 is false; if p1 is true, it does not evaluate p2.
   - Therefore, && is referred to as the conditional or short-circuit AND operator, and || is referred to as the conditional or short-circuit OR operator.

4. Selection statements are used for programming with alternative courses. There are several types of selection statements: if statements, if ... else statements, nested if statements, switch statements, and conditional expressions.
5. The switch statement makes control decisions based on a switch expression of type char, byte, short, or int.
6. The keyword break is optional in a switch statement, but it is normally used at the end of each case in order to terminate the remainder of the switch statement. If the break statement is not present, the next case statement will be executed.

Chapter 4 Mathematical Functions, Characters, and Strings

- Solve mathematics problems by using the methods in the **Math** class, including min/max/pow/sqrt/round/random [a+(int)Math.random() *b)
- Char type represent characters. Java characters use *Unicode*, a 16-bit encoding scheme
- Strings are objects encapsulated in the String class. A string can be constructed using constructors or using a string literal shorthand initializer.
- A String object is immutable; The contents in a immutable object cannot be changed. To improve efficiency and save memory, the JVM stores two literal strings that have the same character sequence in a unique object. This unique object is called an interned string object.
- You can get the length of a string by invoking its length() method, retrieve a character at the specified index in the string using the charAt(index) method, use the concat method to concatenate two strings, or the plus (+) sign to concatenate two or more strings.; use the substring method to obtain a substring from the string; use the toLowerCase and toUpperCase method to convert case.

- use the equals and compareTo methods to compare strings. The equals method returns true if two strings are equal, and false if they are not equal. The compareTo method returns 0, a positive integer, or a negative integer, depending on the Unicode value of each character in the strings.
- The Character class is a wrapper class for a single character. The Character class provides useful static methods to determine whether a character is a letter (isLetter(char)), a digit (isDigit(char)), uppercase (isUpperCase(char)), or lowercase (isLowerCase(char)), and methods such as toLowerCase(char) and toUpperCase (char) .
- Use the printf statement to format the output and a format specifier begins with a percent sign (eg. %md, %ms,%m.nf).
- You can pass strings to the main method from the command line. Strings passed to the main program are stored in args, which is an array of strings. The first string is represented by args[0], and args.length is the number of strings passed.

Chapter 5 Loops

1. There are three types of repetition statements: the while loop, the do-while loop,and the for loop.
   - The while loop and for loop are called *pretest loops because the continuation condition* is checked before the loop body is executed.
   - The do-while loop is called posttest loop because the condition is checked after theloop body is executed.

2. Two keywords, break and continue, can be used in a loop.
   - The break keyword immediately ends the innermost loop, which contains the break.
   - The continue keyword only ends the current iteration.

Chapter 6 Methods

1. The method header specifies the modifiers, return value type, method name, and parameters of the method.
2. A return statement can also be used in a void method for terminating the method and returning to the method's caller. This is useful occasionally for circumventing the normal flow of control in a method.
3. A value-returning method can also be invoked as a statement in Java. In this case, the caller simply ignores the return value.
4. Each time a method is invoked, the system stores parameters and local variables in a space known as a *stack*. *When a method calls another method, the caller's* stack space is kept intact, and new space is created to handle the new method call.When a method finishes its work and returns to its caller, its associated space is released.
5. A method can be overloaded. This means that two methods can have the same name,as long as their method parameter lists differ.

6. A variable declared in a method is called a local variable. The scope of a local variable starts from its declaration and continues to the end of the block that contains the variable. A local variable must be declared and initialized before it is used.

7. *Method abstraction is achieved by separating the use of a method from its implementation.* The client can use a method without knowing how it is implemented. The details of the implementation are encapsulated in the method and hidden from the client who invokes the method. This is known as *information hiding or encapsulation*. Method abstraction modularizes programs in a neat, hierarchical manner.

Chapter 7 Single-Dimensional Arrays

- A variable is declared as an array type using the syntax elementType[] arrayRef-Var or elementType arrayRefVar[]. The style elementType[] arrayRefVar is preferred, although elementType arrayRefVar[] is legal.
- Unlike declarations for primitive data type variables, the declaration of an array variable does not allocate any space in memory for the array. An array variable is not a primitive data type variable. An array variable contains a reference to an array.
- Array element is stored in heap , which is a memory area can be dynamic allocated.
- You cannot assign elements to an array unless it has already been created. You can create an array by using the new operator with the following syntax: new element-Type[arraySize].
- Each element in the array is represented using the syntax arrayRefVar[index]. An index must be an integer or an integer expression.

- After an array is created, its size becomes permanent and can be obtained using arrayRefVar.length. Since the index of an array always begins with 0, the last index is always arrayRefVar.length - 1. An out-of-bounds error will occur if you attempt to reference elements beyond the bounds of an array.
- When an array is created, its elements are assigned the default value of 0 for the numeric primitive data types, '\u0000' for char types, and false for boolean types.
- Java has a shorthand notation, known as the *array initializer, which combines in one* statement declaring an array, creating an array, and initializing, using the syntax: elementType[] arrayRefVar = {value0, value1, ..., valuek}.
- When you pass an array argument to a method, you are actually passing the reference of the array; that is, the called method can modify the elements in the caller's original array

- You can use two simple, intuitive sorting algorithms: selection sort and insertion sort to sort a list.
- The java.util.Arrays class contains various static methods for sorting and searching arrays, comparing arrays, and filling array elements.
- You can use the binarySearch to find the key in a array , which returns -insertionpoint -1 .
    - java.util.Arrays.binarySearch (sorted array)
- You can use the sort method to sort a whole array or a partial array.
    - java.util.Arrays.sort(unsorted array)

Chapter 9 Objects and Classes

- A class is a template for objects. It defines the properties of objects and provides constructors for creating objects and methods for manipulating them.

- A class is also a data type. You can use it to declare object reference variables. An object reference variable that appears to hold an object actually contains a reference to that object. Strictly speaking, an object reference variable and an object are different, but most of the time the distinction can be ignored.

- An object is an instance of a class. You use the new operator to create an object, and the dot (.) operator to access members of that object through its reference variable.

- An instance variable or method belongs to an instance of a class. Its use is associated with individual instances. A static variable is a variable shared by all instances of the same class. A static method is a method that can be invoked without using instances.

- Every instance of a class can access the class's static variables and methods. For clarity, however, it is better to invoke static variables and methods using ClassName.variable and ClassName.method.
- Modifiers specify how the class, method, and data are accessed. A public class, method, or data is accessible to all clients. A default method or data is accessible only inside the same package. A private method or data is accessible only inside the class.
- All parameters are passed to methods using pass-by-value. For a parameter of a primitive type, the actual value is passed; for a parameter of a reference type, the reference for the object is passed.
- A Java array is an object that can contain primitive type values or object type values. When an array of objects is created, its elements are assigned the default value of null.
- For a class to be immutable, it must mark all data fields private and provide no mutator methods and no accessor methods that would return a reference to a mutable data field object .

Chapter 10 Thinking in Objects

- The procedural paradigm focuses on designing methods. The object-oriented paradigm couples data and methods together into objects. Software design using the object-oriented paradigm focuses on objects and operations on objects. The object-oriented approach combines the power of the procedural paradigm with an added dimension that integrates data with operations into objects.

- The scope of instance and static variables is the entire class, regardless of where the variables are declared. Instance and static variables can be declared anywhere in the class. For consistency, they are declared at the beginning of the class.

- The keyword this can be used to refer to the calling object. It can also be used inside a constructor to invoke another constructor of the same class.

- Many Java methods require the use of objects as arguments. Java offers a convenient way to incorporate, or wrap ,a primitive data type into an object(eg.,wrapping int into the Integer class, double into Double class, char into Character class)
  - Integer static methods: static int parseInt(String s) / static Integer valueof(String or int)
  - Integer instance method : String toString()
  - Integer a=100;Integer b=100; a==b
- Note: By default for the values -128 to 127, Integer.valueOf() method will not create a new instance of Integer. It returns a value from its cache.
- Java can automatically convert a primitive type value to its corresponding wrapper object in the context and vice versa. (Boxing and Unboxing)
- A String object is immutable; its contents cannot be changed. To improve. Efficiency and save memory, the JVM stores two literal strings that have the same character sequence in a unique object. This unique object is called an interned string object.
- A regular expression is a string that describes a pattern for matching a set of strings. You can match, replace ,or split a string by specifying a pattern.
- String/String Builder/String Buffer (append/insert/delete)

Chapter 11 Inheritance and Polymorphism

- You can derive a new class from an existing class with extends keyword. This is known as class inheritance.
- The constructors of a superclass are not inherited in the subclass. They can be invoked only from the constructors of the subclasses, using the keyword super.
- Use super : super() or super. method(); / Use this: this. field or this(args)
- A constructor may invoke an overloaded constructor or its superclass's constructor. The call must be the first statement in the constructor. If none of them is invoked explicitly, the compiler puts super() as the first statement in the constructor, which invokes the superclass's no-arg constructor.
- To override a method, the method must be defined in the subclass using the same signature as in its superclass.
- Every class in Java is descended from the java.lang.Object class. If no inheritance is specified when a class is defined, its superclass is Object.

- A class defines a type. A type defined by a subclass is called a *subtype and a type* defined by its superclass is called a *supertype*.
- If a method's parameter type is a superclass (e.g., Object), you may pass an object to this method of any of the parameter's subclasses (e.g., Circle or String). This is known as Ploymorphism.
- When an object (e.g., a Circle object or a String object) is used in the method, the particular implementation of the method of the object that is invoked (e.g., toString) is determined dynamically, also known as dynamic binding.
- Method matching: . The compiler finds a matching method according to method signature at compilation time.
- When invoking an instance method from a reference variable, the *actual type of the* variable decides which implementation of the method is used *at runtime*. *When* accessing a field or a static method, the *declared type of the reference variable* decides which method is used *at compile time*.

- You can use obj instanceof a class to test whether an object is an instance of a class.
- You can use the protected modifier to prevent the data and methods from being accessed by nonsubclasses from a different package.
- You can use the final modifier to indicate that a class is final and cannot be a parent class and to indicate that a method is final and cannot be overridden.
- A method declared static cannot be overridden but can be re-declared.
- If a method cannot be inherited ,then it cannot be overridden(private/final)
- java.util.ArrayList Class can be used to create an object to store a list of objects.

Chapter 12 Exception Handling and Text IO

- Exception handling enables a method to throw an exception to its caller.
- A Java exception is an instance of a class derived from java.lang.Throwable. Java provides a number of predefined exception classes, such as Error, Exception, RuntimeException, ClassNotFoundException, IOException. NullPointerException, and ArithmeticException. You can also define your own exception class by extending Exception.
- Exceptions occur during the execution of a method. RuntimeException and Error are unchecked exceptions; all other exceptions are checked.
- When declaring a method, you have to declare a checked exception if the method might throw it, thus telling the compiler what can go wrong.
- The keyword for declaring an exception is throws, and the keyword for throwing an exception is throw.

- To invoke the method that declares checked exceptions, you must enclose the method call in a try statement. When an exception occurs during the execution of the method, the catch block catches and handles the exception.

- If an exception is not caught in the current method, it is passed to its caller. The process is repeated until the exception is caught or passed to the main method.

- Various exception classes can be derived from a common superclass. If a catch block catches the exception objects of a superclass, it can also catch all the exception objects of the subclasses of that superclass.

- The order in which exceptions are specified in a catch block is important. A compile error will result if you do not specify an exception object of a class before an exception object of the superclass of that class.

- The code in the finally block is executed under all circumstances, regardless of whether an exception occurs in the try block or is caught.

- Exception handling separates error-handling code from normal programming tasks, thus making programs easier to read and to modify.

- Exception handling should not be used to replace simple tests. You should test simple exceptions whenever possible, and reserve exception handling for dealing with situations that cannot be handled with if statements.

- The File class is used to obtain file properties and manipulate files. It does not contain the methods for creating a file or for reading/writing data from/to a file.

- You can use Scanner to read string and primitive data values from a text file and use PrintWriter to create a file and write data to a text file. You can use try-with-resources to close resources automatically.

Chapter 13 Abstract Classes and Interfaces

- Abstract classes are like regular classes with data and methods, but you cannot create instances of abstract classes using the new operator.
- An abstract method cannot be contained in a nonabstract class. If a subclass of an abstract superclass does not implement all the inherited abstract methods of the superclass,the subclass must be defined abstract.
- A class that contains abstract methods must be abstract. However, it is possible to define an abstract class that contains no abstract methods.
- A subclass can be abstract even if its superclass is concrete.

- An interface is a classlike construct that contains only constants and abstract methods. In many ways, an interface is similar to an abstract class, but an abstract class can contain constants and abstract methods as well as variables and concrete methods.

- An interface is treated like a special class in Java. Each interface is compiled into a separate bytecode file, just like a regular class.

- The java.lang.Comparable interface defines the compareTo method. Many classes in the Java library implement Comparable.
- The java.lang.Cloneable is a marker interface does not contain constants or methods. It is used to denote that a class possesses certain desirable properties.(Shallow copy / deep copy)
- A class can extend only one superclass but can implement one or more interfaces.

- An interface can extend one or more interfaces.

Chapter 14 Java FX Basics

- JavaFX is the new framework for developing rich Internet applications. JavaFx completely replaces Swing and AWT.
- A main JavaFX class must extend javafx.application.Application and implement the start method. The primary stage is automatically created by the JVM and passed to the start method.
- A stage is a window for displaying a scene. You can add nodes to a scene. Panes,controls, and shapes are nodes. Panes can be used as the containers for nodes.
- JavaFX provides many types of panes for automatically laying out nodes in a desired location and size. The Pane is the base class for all panes. It contains the getChildren() method to return an ObservableList. You can use ObservableList's add(node) and addAll(node1,node2,…) methods for adding nodes into a pane.
- An Hbox lays out its children in a single horizontal row. A Vbox lays out its children in a Single vertical column.
- Pane /Circle/ TextField/Button

Chapter 15 Event-driven Programming

- The root class of the event classes is javafx.event.Event. The subclasses of Event deal with special types of events, such as action events, window events, component events, mouse events, and key events. If a component can fire an event, any subclass of the component can fire the same type of event.

- The handler object's class must implement the corresponding handler interface. JavaFX provides a handler interface for every event class T EventHandler<T extends Event>. The handler interface contains the method(s), known as the *handler(T e), which process the events.*

- The Handler object must be registered by the source object. Registration methods depend on the event type. For Action Event, the method is setOnAction.
- Java Event Delegation Model
  - Event/Event source /Event handler(Listener)

- An *inner class, or nested class, is defined within the scope of another class. An inner* class can reference the data and methods defined in the outer class in which it nests, so you need not pass the reference of the outer class to the constructor of the inner class.

  (OuterClassName$innerclassName.class.)

- An anonymous inner class must always extend a superclass or implement an interface, but it cannot have an explicit extends or implements clause.
- An anonymous inner class is compiled into a class named OuterClassName$n.class.
- We can simplify event handing using lambda expressions. lambda expressions can be viewed as an anonymous method with a concise syntax.
  - (type1 param1, type2 param2, ...) -> expression
  - (type1 param1, type2 param2, ...) -> { statements; }

Exam

- Multiple choice (30pts, each 2 pts)
- True or False (10pts)
- Short Answer (15 pts, each 5pts)
- Read the program and write the output (15pts,each 5pts)
- Programming problem (30pts, 2 questions)

Multiple choice

- 1. To prevent a class from being instantiated, _____

A. use the private modifier on the constructor.

B. use the public modifier on the constructor.

C. don't use any modifiers on the constructor.

D. use the static modifier on the constructor.

2. When you return an array from a method, the method returns _____.

A. a copy of the array

B. a copy of the first element

C. the reference of the array

D. the length of the array

3. The _____ method parses a string s to an int value.

A. integer.parseInt(s);
B. Integer.parseInt(s);
C. integer.parseInteger(s);
D. Integer.parseInteger(s);

Multiple choice

- 1. To prevent a class from being instantiated, ___A_____
  - A. use the private modifier on the constructor.
  - B. use the public modifier on the constructor.
  - C. don't use any modifiers on the constructor.
  - D. use the static modifier on the constructor.

2. When you return an array from a method, the method returns ____C_____.

A. a copy of the array

B. a copy of the first element

C. the reference of the array

D. the length of the array

3. The ____B_____ method parses a string s to an int value.

A. integer.parseInt(s);
B. Integer.parseInt(s);
C. integer.parseInteger(s);
D. Integer.parseInteger(s);

4. Analyze the following code.

```
public class Test {
    public static void main(String[ ] args) {
        System.out.println(m(2));
    }
    public static int m(int num) {
        return num;
    }
    public static void m(int num) {
        System.out.println(num);
    }
}
```

A) The program runs and prints 2 twice.

B) The program has a compile error because the two methods m have the same signature.

C) The program runs and prints 2 once.

D) The program has a compile error because the second m method is defined, but not invoked in the main method.

4. Analyze the following code. **B**

```
public class Test {
    public static void main(String[ ] args) {
        System.out.println(m(2));
    }
    public static int m(int num) {
        return num;
    }
    public static void m(int num) {
        System.out.println(num);
    }
}
```

A) The program runs and prints 2 twice.

B) The program has a compile error because the two methods m have the same signature.

C) The program runs and prints 2 once.

D) The program has a compile error because the second m method is defined, but not invoked in the main method.

Short Answers

- What is the purpose of declaring exceptions? How do you declare an exception, and where? Can you declare multiple exceptions in a method header?
- Suppose that **statement2 causes an exception in the following try-catch** block:

```
try {
    statement1;
    statement2;
    statement3;
}
catch (Exception1 ex1) {
    }
catch (Exception2 ex2) {
    }
statement4;
```

Answer the following questions:

- Will **statement3 be executed?**
- If the exception is not caught, will **statement4 be executed?**
- If the exception is caught in the **catch block, will statement4 be executed?**
- If the exception is passed to the caller, will **statement4 be executed?**

Short Answers

- What is the purpose of declaring exceptions? How do you declare an exception, and where? Can you declare multiple exceptions in a method header?
- Suppose that **statement2 causes an exception in the following try-catch** block:

```
try {
    statement1;
    statement2;
    statement3;
}
catch (Exception1 ex1) {
    }
catch (Exception2 ex2) {
    }
statement4;
```

Answer the following questions:

- Will **statement3 be executed?** **no**
- If the exception is not caught, will **statement4 be executed?** **no**
- If the exception is caught in the **catch block, will statement4 be executed?** **yes**
- If the exception is passed to the caller, will **statement4 be executed?** **no**

True of false

- When invoking a constructor from a subclass, its superclass's no-arg constructor is always invoked.

- Read the program and write the output

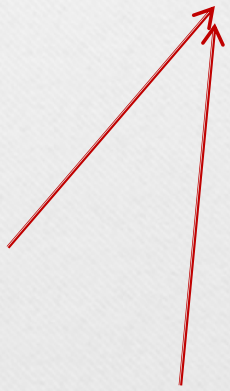
```
1.  interface A {
2.        }
3.  class C {
4.        }
5.  class B extends D implements A {
6.        }
7.  public class Test {
8.      public static void main(String[ ] args) {
9.          B b = new B();
10.         if (b instanceof A)
11.                 System.out.println("b is an instance of A");
12.         if (b instanceof C)
13.                 System.out.println("b is an instance of C");
14.     }
15. }
16. class D extends C {
17.       }
```

True of false

- When invoking a constructor from a subclass, its superclass's no-arg constructor is always invoked. （F）
- There is one error in this program. Please find the error statement, and correct it.
- Read the program and write the output

```
1.  interface A {
2.          }
3.  class C {
4.          }
5.  class B extends D implements A {
6.           }
7.  public class Test {
8.       public static void main(String[ ] args) {
9.            B b = new B();
10.           if (b instanceof A)
11.                 System.out.println("b is an instance of A");
12.           if (b instanceof C)
13.                 System.out.println("b is an instance of C");
14.      }
15. }
16. class D extends C {
17.          }
```

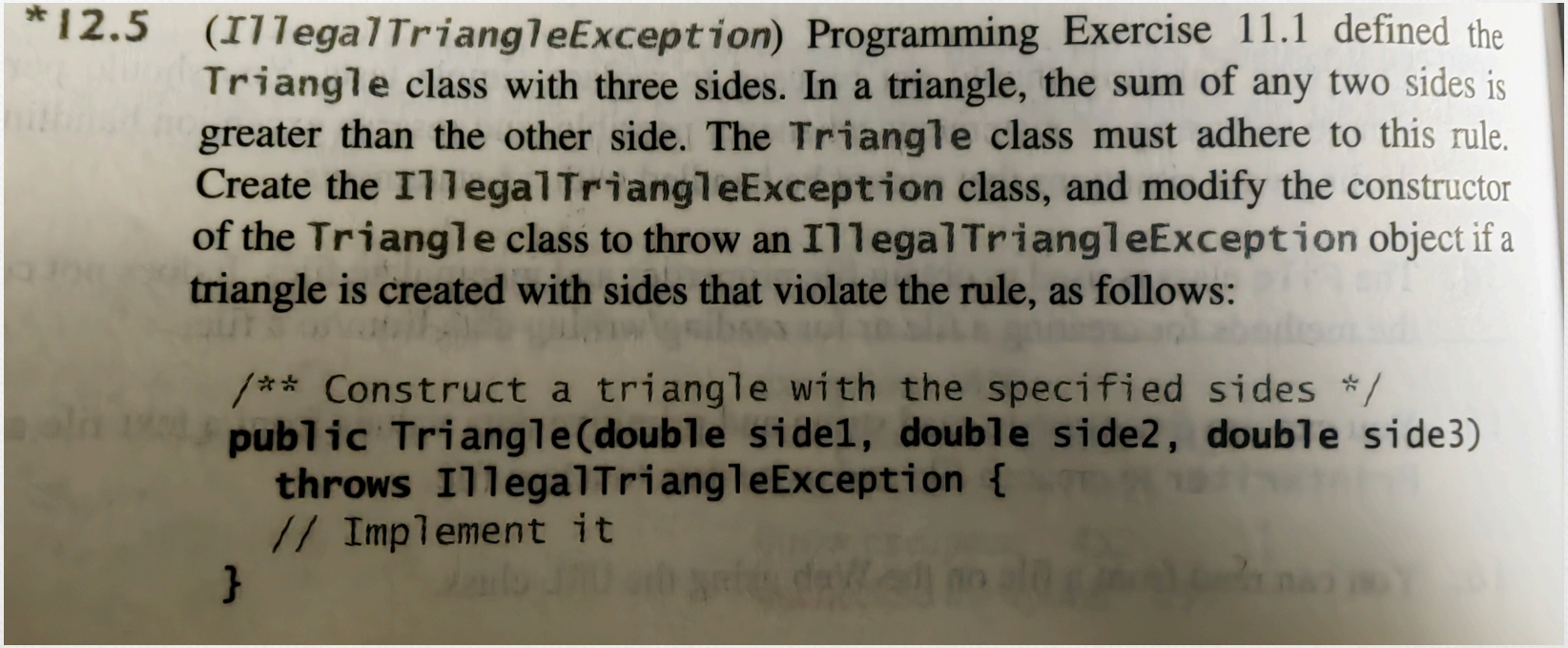
b is an instance of A
b is an instance of C



Programming problem

Exercises

- Chapter 12 12.5

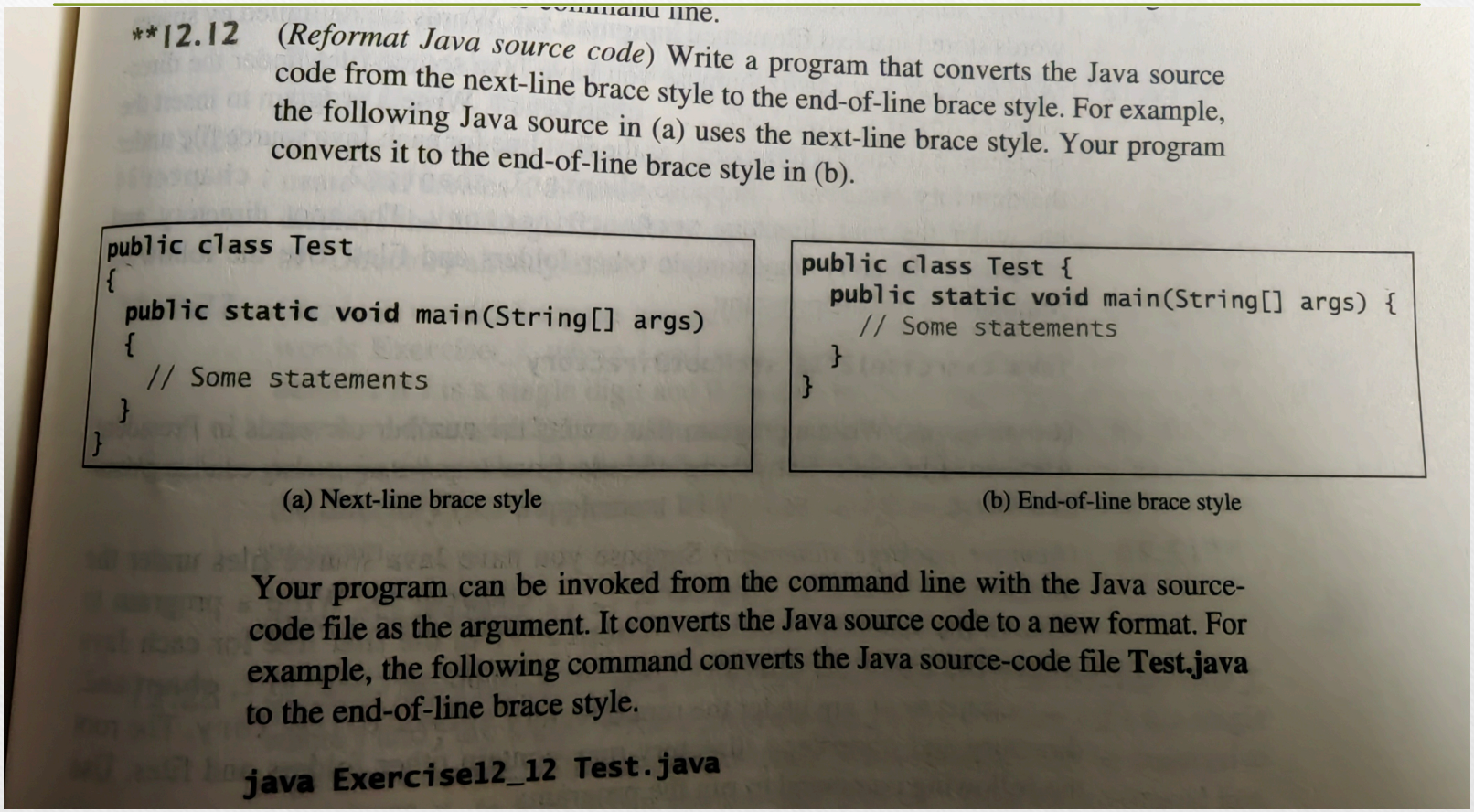


*12.5 (*IllegalTriangleException*) Programming Exercise 11.1 defined the `Triangle` class with three sides. In a triangle, the sum of any two sides is greater than the other side. The `Triangle` class must adhere to this rule. Create the `IllegalTriangleException` class, and modify the constructor of the `Triangle` class to throw an `IllegalTriangleException` object if a triangle is created with sides that violate the rule, as follows:

```
/** Construct a triangle with the specified sides */
public Triangle(double side1, double side2, double side3)
  throws IllegalTriangleException {
  // Implement it
}
```

Exercises

- Chapter 12 12.12



command line.

**12.12 (*Reformat Java source code*) Write a program that converts the Java source code from the next-line brace style to the end-of-line brace style. For example, the following Java source in (a) uses the next-line brace style. Your program converts it to the end-of-line brace style in (b).

```
public class Test
{
  public static void main(String[] args)
  {
    // Some statements
  }
}
```

(a) Next-line brace style

```
public class Test {
  public static void main(String[] args) {
    // Some statements
  }
}
```

(b) End-of-line brace style

Your program can be invoked from the command line with the Java source-code file as the argument. It converts the Java source code to a new format. For example, the following command converts the Java source-code file **Test.java** to the end-of-line brace style.

```
java Exercise12_12 Test.java
```