Object-Oriented Programming 2022/23
doc. Ing. Valentino Vranić, PhD., ÚISI FIIT STU
Exam – May 23, 2023

Last name:

Name:

1 b	
2 b	
3 b	

1	
2	
3	
4	
5	
6	
7	
8	
9	
10	
11	
12	

The exam can take up to 70 minutes.

Write the answers to questions 1–12 into the table. With these questions, only the answers in the table will be considered (without the work out). An answer must be unambiguous and readable, otherwise it will be marked with 0 points.

In multiple-choice questions only one choice is correct – write into the table only the letter by which the answer you choose is marked with.

Write the answer to question 13 exclusively on the paper with the question text.

No damaged paper will be accepted.

1. (1 b) In Java, an analogous mechanism to C++ templates

(a) are anonymous classes
(b) RTTI
(c) do not exist
(d) are interfaces
(e) are generics

2. (1 b) Java supports persistence by

(a) radialization
(b) serialization
(c) aggregation
(d) modularization
(e) synchronization

3. (1 b) Is it possible in AspectJ to influence the program behavior by using only inter-type declarations (without using RTTI)?

(a) yes, but not setting the attributes
(b) yes, but only that of static methods
(c) no
(d) yes
(e) yes, but only that of non-static methods

4. (1 b) Generics in Java enables collections to

(a) be specialized for any data type during instantiation
(b) be specialized for any data type during inheritance
(c) store a greater number of objects
(d) perform faster
(e) be automatically saved to the disk

5. (1 b) Classes in the C++ language have their roots in the mechanism of

(a) struct
(b) union
(c) include
(d) template
(e) define

6. (1 b) In C#, events

(a) can't be implemented
(b) are the same as the delegate mechanism
(c) exist as a language mechanism, but can't be used outside a GUI
(d) are not a language mechanism
(e) exist as a language mechanism and can be used outside a GUI

7. (2 b) Extensibility of a class with other operations (methods) should be provided, but so that the adding of operations wouldn't require changing its code. What design pattern would you use?

(a) Observer
(b) Composite
(c) Strategy
(d) Visitor
(e) MVC

8. (2 b) The following program in Java is given:

```
class Zero extends Thread {
    C c;
    public Zero(C c) {
        this.c = c;
    }
    public synchronized void run() {
        for (int i = 0; i < 9999; i++)
            c.zero();
    }
}
class One extends Thread {
    C c;
    public One(C c) {
        this.c = c;
    }
    public synchronized void run() {
        for (int i = 0; i < 9999; i++)
            c.one();
    }
}
class C {
    private int d1 = 0, d2 = 1;
    public void zero() {
        if (d1 != d2)
            System.out.println(0);
        d1 = 0;
        d2 = 0;
    }

    public void one() {
        if (d1 != d2)
            System.out.println(1);
        d1 = 1;
        d2 = 1;
    }
    public static synchronized void main(String[] args) {
        C c = new C();
        new Zero(c).start();
        new One(c).start();
    }
}
```

The output of this program will be

(a) nothing or zeroes and ones in a varying number and alternation
(b) zeroes and ones ten thousand times in a varying number and alternation
(c) zeroes and ones ten thousand times in a regular alternation
(d) nothing
(e) zeroes and ones in a varying number and alternation

9. (2 b) What all makes the output of the System.out.print() statements of the following program in Java (until its successful or unsuccessful ending)?

```
class E extends Exception {}

class M {
   public void m(char c) throws E {
      if (c == 'a')
         System.out.print("A");
      else
         throw new E();
   }

   public void f(char c) throws E {
      System.out.print("F");

      try {
         m(c);
      } catch (E e) {
         throw e;
      } finally {
         System.out.print("!");
      }
   }

   public static void main(String[] args) throws E {
      new M().f('b');
      new M().f('a');
      new M().f('b');
   }
}
```

10. (2 b) A game in Java includes the following code:

```
class Fairy {
   private int energy;
   private int lives;
   ...
   public void energyToLives() {
      acquiredLives = energy/100;
      lives += acquiredLives;
      energy -= acquiredLives * 100;
      MainWindow.window.numberOfLives.setText(
         Integer.toString(lives));
   }
   ...
}
```

The main problem in this code with respect of object-oriented design is that

(a) the code for transforming energy to lives isn't a part of the game window
(b) the attributes are **private** and they won't be accessible by derived classes
(c) the method changes two attributes
(d) the internal logic is being mixed with the user interface
(e) the code for transforming energy to lives isn't a part of the corresponding listener (handler)

11. (3 b) What is the output of the execution of the following program in Java?

```
class A {
   public void x() {
      System.out.print("Ax");
   }
   public static void y() {
      System.out.print("Ay");
   }
}
class B extends A {
   public void x() {
      super.x();
      System.out.print("Bx");
   }
   public static void y() {
      A.y();
      System.out.print("By");
   }
}
class C extends B {
   public void x() {
      System.out.print("Cx");
   }
   public static void y() {
      System.out.print("Cy");
   }
}
class M {
   public static void main(String[] args) {
      B o1 = new C();
      A o2 = new B();
      B o3 = new B();
      C o4 = new C();
      A o5 = new B();

      ((C) o1).x();
      ((C) o1).y();
      System.out.print(" ");

      o2.x();
      o2.y();
      System.out.print(" ");

      o3.x();
      o3.y();
      System.out.print(" ");

      ((B) o4).x();
      ((B) o4).y();
      System.out.print(" ");

      ((A) o5).x();
      ((A) o5).y();
   }
}
```

12. (3 b) The class that represents a special document is devised from the class that represents a general document. The method for identifying the general document signatory does not guarantee returning the name of the signatory because a general document may not have one. By this, preconditions and postconditions of these methods weaken, strengthen, or remain the same? Is Liskov substitution principle (LSP) preserved by this? Is it correct to use inheritance in this case?

Answer in this form: *preconditions / postconditions / LSP / inheritance*. Replace the items *postconditions* and *preconditions* with the one of the following possibilities: *weaken*, *strengthen*, or *remain the same*. Replace the item *LSP* with the one of the following possibilities: *preserved* or *not preserved*. Replace the item *inheritance* with the one of the following possibilities: *yes* or *no*.

Last name:

Name:

13. (10 b) In an urbanistic simulation program, different decorations occur with respect to a city. For the time being, these are colored lights and fountains, but in future, other kinds will be added. In the city, energy is being saved and the decorations indicate when this is necessary to its inhabitants. Thus, colored lights shine blue, when the city instant consumption is under 50% of the long-term average, blue, when the city instant consumption is between 50% and 70% of the long-term average, a red, when the city instant consumption is above 70% of the long-term average. Fountains operate only when the city instant consumption is under 60% of the long-term average. The implementation of calculating the long-term average and instant consumption is not the subject of this task.

Design and implement in Java the corresponding object-oriented solution that takes into account the principles of object-oriented programming. In this, apply the most appropriate design pattern among Strategy, Observer, Visitor, and Composite.

Provide the basic design as a UML class diagram sketch containing the most important relationships, operations, and attributes. In this, take into account the design pattern. The visibility of the elements does not have to be introduced.

In implementation, focus on the application logic – the user interface is not the subject of the question. Also, the algorithms applied do not have to be optimal.

Explicitly identify the elements that model and implement the roles of the design pattern applied and explain why did you apply this design pattern. Present an example of use in which you create the corresponding objects and start off their interaction.

The question would be marked according to the following key:

- providing the basic functionality – 4 b
- a design with respect to the open-closed principle and appropriate use of encapsulation – 6 b

Object-Oriented Programming 2022/23
doc. Ing. Valentino Vranić, PhD., ÚISI FIIT STU
Exam – May 23, 2023
30

1 e

2 b

3 c

4 a

5 a

6 e

7 d

8 a

9 F!

10 d

11 remain the same / strengthen / preserved / yes

12 CxCy AxBxAy AxBxAyBy CxAyBy AxBxAy

13 It's appropriate to apply the Observer pattern. A city and the energy consumption data as such are represented by an abstract class or interface and play the Subject role. From this class or interface, the only concrete city type is derived. Decorations are represented by an abstract class or interface and play the Observer role. From this class or interface, concrete types of decoration are derived. In the method which updates the instant city consumption, the notification of all registered decorations is activated, which in turn update their state.