Objektovo-orientované programovanie 2023/24 **A**

prof. Ing. Valentino Vranić, PhD., ÚISI FIIT STU
Exam – June 11, 2024

Priezvisko:

Meno:

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) Multiple inheritance in C++

(a) serves to reduce the number of classes
(b) replaces the friend mechanism
(c) may contribute to a better separation of concerns
(d) is the same as repeated inheritance
(e) represents an alternative to virtual functions

2. (1 b) The open–closed principle says that

(a) every data stream needs to be closed after opening
(b) code should be open for change, but closed for extension
(c) code should be both open and closed for editing
(d) every data stream except standard input and output must also be closed after opening
(e) code should be open for extension, but closed for changes

3. (1 b) In C++, using the special keyword, overriding

(a) can be disabled if not needed
(b) must be activated if necessary
(c) must be activated if necessary, and can then be deactivated
(d) must be disabled if not needed
(e) must be activated if necessary, and subsequently deactivated

4. (1 b) The fact that an object of another class can be used instead of an object of a given class means that between these classes there is a relationship of

(a) aggregation
(b) integration
(c) segregation
(d) inheritance
(e) genericity

5. (1 b) In the C# language, a delegate represents

(a) a function of corresponding parameter types and return value
(b) a pointer to an object
(c) a property of the corresponding type
(d) any function
(e) any property

6. (1 b) The expression call(abc∗(..)) in AspectJ means

(a) calling the call() method before all methods whose name starts with abc
(b) catch the call of all methods whose name starts with abc
(c) calling the first method whose name starts with abc
(d) calling all methods whose name starts with abc
(e) catching the first call to a method whose name starts with abc

7. (2 b) The following program in Java is given:

```
class C implements Runnable {
   M m;

   public C(M m) {
      this.m = m;
   }
   public void run() {
      for (int i = 0; i < 100000; i++) {
         m.p1();
         m.p2();
         m.p3();
      }
   }
}

public class M {
   private int a = 1, b = 1;

   public void p1() {
      if (a != b)
         System.out.print("1");
      a = 1;
      b = 1;
   }

   public void p2() {
      if (a != b)
         System.out.print("2");
      a = 2;
      b = 2;
   }

   public void p3() {
      synchronized(this) {
         if (a != b)
            System.out.print("3");
         a = 3;
         b = 3;
      }
   }

   public static void main(String[] args) {
      M m = new M();
      new Thread(new C(m)).start();
      new Thread(new C(m)).start();
   }
}
```

At which methods the modifier **synchronized** must be added to guarantee that nothing is ever printed (specify them in this notation: `Class.method()`)?

8. (2 b) For which design pattern the notification of certain objects when the state of another object changes is indicative?

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

9. (2 b) What is all printed out by the commands System.out.print() after running the following Java program (up to its successful or unsuccessful completion)?

```
public class E {
   public void c(int n) throws Exception {
      if (n == 0)
         throw new Exception();
   }

   public void m(int n) {
      try {
         c(n);
      } catch (Exception e) {
         System.out.print("E");
      } finally {
          System.out.print(n);
      }
   }

   public static void main(String[] args) {
      E e = new E();
      e.m(2);
      e.m(0);
      e.m(2);
   }
}
```

10. (2 b) The graphical user interface of a computer game is created using the JavaFX framework. It also includes the t button, in connection with which the following code occurs in the game:

```
t.setOnAction(e −> {
   if (player.hasShield())
      player.setEnergy(player.getEnergy() − 1);
   else
      player.setEnergy(player.getEnergy() − 2);
});
```

The primary problem with this code from an object-oriented design perspective is that

(a) application logic occurs in the user interface
(b) the event handler was not implemented by an anonymous class
(c) polymorphism was not used
(d) encapsulation was not used
(e) it does not contain a comment

11. (3 b) What is all printed out by the commands System.out.print() after running the following Java program?

```
class C {
   public void f() {
      System.out.print("C");
   }
}
class D extends C {
   public void f() {
      super.f();
      System.out.print("D");
   }
}
class E extends D {
   public void f() {
      super.f();
      System.out.print("E");
   }
}
class F extends E {
   public void f() {
      super.f();
      System.out.print("F");
   }
}
public class M {
   public static void main(String[] args) {
      F o1 = new F();
      C o2 = new E();
      E o3 = new E();
      D o4 = new F();

      ((E) o1).f();
      System.out.print(" ");

      ((D) o2).f();
      System.out.print(" ");

      ((C) o3).f();
      System.out.print(" ");

      ((C) o4).f();
      System.out.print(" ");
    }
}
```

12. (3 b) The method guarantees that it will return a whole positive number after the calculation. The method that overrides it returns a whole number that can also be negative.

By this, preconditions and postconditions of these methods are weakened, strengthened, 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: *weakened*, *strengthened*, 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*.

Priezvisko:

Meno:

13. (10 b) A virtual space consists of connected cells. The cell can be indivisible or divisible. A divisible cell may contain other indivisible and divisible cells. Cells can be connected regardless of whether they are indivisible or divisible. For each cell, it is possible to get a list of cells that can be accessed from this cell, which are the cells to which the given cell is connected directly and – in case of a divisible cell – the cells, which the given cell consists of (the first level).

Provide the basic design as a UML class diagram sketch containing the most important relationships, operations, and attributes. In this, apply the most appropriate design pattern among Strategy, Observer, Visitor, and Composite. Explicitly identify the elements that model and implement the roles of the design pattern applied. The visibility of the elements does not have to be introduced.

Provide the corresponding implementation in Java (the code itself). In implementation, focus on the application logic – the user interface is not the subject of the question. Take into account the principles of object-oriented programming. Provide an example of use in which you create the corresponding objects and start off their interaction.

The question will be marked according to the following key:

- providing the basic functionality – 4 b
- quality and flexibility of the object-oriented design – 6 b

Objektovo-orientované programovanie 2023/24 $\boxed{\textbf{A}}$
prof. Ing. Valentino Vranić, PhD., ÚISI FIIT STU
Exam – June 11, 2024
30

1 c

2 e

3 b

4 d

5 a

6 b

7 M.p1() a M.p2()

8 c

9 2E02

10 a

11 CDEF CDE CDE CDEF (akceptovaná je aj odpoveď bez medzier)

12 remain the same / weakened / not preserved / no

13 The Composite pattern should have been applied. A cell as such would be in the role of Component (interface or abstract class), a divisible cell in the role of Composite, and an indivisible cell in the role of Leaf (classes derived from the class representing the cell). The operation of listing or returning connections should have been implemented in such a way that it could be applied transparently over both divisible and indivisible cells (it should have been present in the cell, and overridden in divisible and indivisible cells).

Both divisable and indivisable cells should contain evidence (e.g., a list or array) of the cells they are associated with. In addition, a divisable cell should contain a record of the cells it consists of. The list or return connection operation for an indivisible cell should return only the cells the given cell is connected to, while for a divisible cell, the cells of which it consists should also be added.