Object-Oriented Programming 2021/22 $\boxed{\textbf{A}}$
doc. Ing. Valentino Vranić, PhD., ÚISI FIIT STU
Exam – June 1, 2022

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) A delegate in the C# language is a form of a pointer to

(a) a function
(b) object
(c) class
(d) macros
(e) templates

2. (1 b) In Java, an analogous mechanism to C++ templates

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

3. (1 b) In C++, a new object can be created

(a) only out of classes that do not contain virtual functions
(b) not out of a class, but only out of a template
(c) exclusively by defining a variable of the type of a given class
(d) exclusively by the **new** operator as in Java
(e) by the **new** operator as in Java or by defining a variable of the type of a given class

4. (1 b) Serialization in Java serves to

(a) protect data
(b) store source code
(c) aggregate data
(d) synchronize data
(e) store data

5. (1 b) To what is in Java implemented by attributes with accompanying set and get methods, in the C# language corresponds the mechanism of

(a) var
(b) sealed
(c) property
(d) delegate
(e) event

6. (1 b) Is it possible in AspectJ to influence the program behavior by using only inter-type declarations (without using RTTI)?

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

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

```
class C implements Runnable {
   M m;
   public C(M m) {
      this.m = m;
   }
   public synchronized void run() {
      for (int i = 0; i < 100000; i++) {
            m.a();
            m.c();
      }
   }
}

public class M {
   private char x = 'a', y = 'a';

   public synchronized void a() {
      if (x != y)
         System.out.println("a");
      x = 'a';
      y = 'a';
   }

   public synchronized void b() {
      synchronized(this) {
         if (x != y)
            System.out.println("b");
         x = 'b';
         y = 'b';
      }
   }

   public synchronized void c() {
         synchronized(this) {
            if (x != y)
               System.out.println("c");
            x = 'c';
            y = 'c';
         }
      }

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

At which methods it is possible to remove the **synchronized** modifier so that it remains guaranteed that nothing ever appears in the output (introduce them in this form: `Class.method()`)?

8. (2 b) New ways of realizing a certain functionality can occur in a given context. Which design pattern would you use?

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

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)?

```
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(0);
      e.m(2);
      e.m(0);
   }
}
```

10. (2 b) Does the MVC pattern assures separating the inner logic from user interface? If so, which role of this pattern predominantly implemnts the inner logic, and which one the user interface?

Answer in this form: *separation / inner logic / user interface*. Replace the items *separation* nahraďte jednou z možností *yes* or *no* according to that whether you think that the MVC pattern assures separating the inner logic from user interface or not. If you answered *yes*, replace the items *inner logic* and *user interface* with the corresponding roles of this pattern.

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

```
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) {
      E o1 = new E();
      D o2 = new E();
      C o3 = new F();
      F o4 = new F();
      C o5 = new D();

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

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

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

      ((E) o4).f();
      System.out.print(" ");

      ((C) o5).f();
   }
}
```

12. (3 b) The class that represents a spaceship special with automatic flight control is devised from the class that represents a basic spaceship. The method for assigning the spaceship captain, one parameter of which is exactly the captain, is in the spaceship special with automatic flight control overridden and allows for a captain of a lower rank than with the basic spaceship. By this, postconditions and preconditions of these methods weaken, strengthen, or remain the same? Is Liskov substitution principle (LSP) preserved by this?

Answer in this form: *postconditions / preconditions / 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 a simplified computer representation a city consists of building objects which represent any kind of building or independent units within these buildings. Each such independent unit can be divided into further independent units and rooms that are not divided further.

A user may enter a building object by which the system prints a list of the identifiers of the building objects and rooms out of which this building object consists of. In case a user enters a room, the system prints only the identifier of this room.

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 2021/22 **A**
doc. Ing. Valentino Vranić, PhD., ÚISI FIIT STU
Exam – June 1, 2022
30

1 a

2 d

3 e

4 e

5 c

6 a

7 C.run(), M.b(), M.c(), M.main()

8 d

9 E02E0

10 áno / Model / View

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

12 nemenia sa / zoslabujú sa / dodržaný / áno

13 Vhodný je vzor Composite. Stavebné objekty by boli v role Composite, a miestnosti v role Leaf. Obidva druhy objektov by boli zastrešené rolou Component.