Object-Oriented Programming 20018/19 **A**

doc. Ing. Valentino Vranić, PhD., ÚISI FIIT STU

Exam – June 7, 2019

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.

A damaged paper will not be accepted.

1. (1 b) In the C++ language, by overloading an operator

(a) the meaning of this operator changes for all types of operands
(b) the meaning of this operator changes for a given type of operands
(c) the meaning of this operator doesn't change at all
(d) the meaning of this operator changes by an intervention in the compiler
(e) the meaning of this operator changes by an intervention in the linker

2. (1 b) In the C++ language, a destructor

(a) must be introduced in all classes
(b) is never introduced
(c) must be introduced in classes that are derived from several other classes
(d) must be introduced in classes in which memory is being allocated when objects are created
(e) must be introduced in classes that contain virtual methods

3. (1 b) The C# language simplifies providing the encapsulation by

(a) delegates
(b) events
(c) properties
(d) templates
(e) macros

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) In Java, a method being generic means that

(a) it has no body
(b) it has no parameters nor a return value
(c) it has at least one parameter or return value of the Object type
(d) it doesn't implement all the details, but a part of its code is generated automatically according to parameters
(e) it has the type of at least one of its parameters represented by a variable

6. (1 b) In the AspectJ language, without changing the code of a method it is possible to

(a) fully manage method execution
(b) ensure executing code before, after, or instead of a method, but not executing a method conditionally
(c) ensure executing code before or after a method, but not instead of a method
(d) add new methods, but not to affect the behavior of existing methods
(e) add new attributes, but not methods

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

```
class S1 extends Thread {
    C c;
    public S1(C c) {
        this.c = c;
    }
    public synchronized void run() {
        synchronized(this) {
            for (int i = 0; i < 10000; i++)
                c.s1();
        }
    }
}

class S2 implements Runnable {
    C c;
    public S2(C c) {
        this.c = c;
    }
    public synchronized void run() {
        for (int i = 0; i < 10000; i++)
            c.s2();
    }
}

class C {
    private char o1 = 0, o2 = 0;

    public void s1() {
        if (o1 != o2)
            System.out.println("s1");
        o1 = 1;
        o2 = 1;
    }

    public synchronized void s2() {
        synchronized(this) {
            if (o1 != o2)
                System.out.println("s2");
            o1 = 2;
            o2 = 2;
        }
    }

    public static synchronized void main(String[] args) {
        C c = new C();
        new S1(c).start();
        new Thread(new S2(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) It is necessary to ensure that the objects that behave according to changes in the state of a given object know about these changes and that such objects could be added without the necessity to modify this object.

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

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 C {
    public void f(int n) throws E {
        if (n < 1)
            throw new E();
    }

    public void g(int n) {
        System.out.print(n);

        try {
            f(n);
        } catch (E e) {
            System.out.print("E");
        } finally {
            System.out.print("F ");
        }
    }

    public static void main(String[] args) {
        new C().g(1);
        new C().g(0);
        new C().g(1);
        new C().g(0);
    }
}
```

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

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

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

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

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) {
        A o1 = new B();
        C o2 = new C();
        A o3 = new A();
        B o4 = new B();
        B o5 = new C();

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

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

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

        o4.x();
        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 signing the document, one parameter of which is a signatory, is in the special document overridden and requires that the signatory be from the list of authorized signatories. 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*. Items *postconditions* and *preconditions* replace with the one of the following possibilities: *weaken*, *strengthen*, or *remain the same*. Items *LSP* replace with the one of the following possibilities: *preserved* or *not preserved*.

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. Visibility is not required to be provided.

In implementation, focus on the application logic – GUI 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 2018/19 **A**
doc. Ing. Valentino Vranić, PhD., ÚISI FIIT STU
Exam – June 7, 2019
30

1 b

2 d

3 c

4 e

5 e

6 a

7 C.main(), C.s2(), S1.run(), S2.run()
(keďže pri metóde C.s1() omylom nebol uvedený modifikátor **synchronized**, akceptovaná je aj odpoveď typu „nedá sa dosiahnuť“)

8 b

9 1F 0EF 1F 0EF (akceptovaná je aj odpoveď bez medzier)

10 a

11 AxBxAyBy CxAy AxAy AxBxAyBy CxAy (akceptovaná je aj odpoveď bez medzier)

12 nemenia sa / zosilňujú sa / nedodržaný

V poslednej otázke mal byť aplikovaný vzor Composite. Stavebné objekty by boli v role Composite, a miestnosti v role Leaf. Obidva druhy objektov by boli zastrešené rolou Component.