DELFT UNIVERSITY OF TECHNOLOGY
Faculty of Electrical Engineering, Mathematics and Computer Science

TI1206 Object Oriented Programming (multiple choice part)
January 9th, 2017, 18:30-20:30

Exam created by dr. A. Zaidman and checked by dr. G. Gousios

During this exam you are *not allowed* to make use of aids like your book or the slides. Digital devices such as phone, tablet, … stay in your bag and are turned off/silent.

Further information:

- Each question has 1 correct answer.
- There are 25 questions, each question has the same weight for determining your grade.
- This exam = 9 pages (including appendix)
- Fill in your answers on a separate form that will be processed automatically
  - Ask for a new form when you want to correct your answers on the form
  - Do not forget to fill in your name and student number on this form. Also, please sign it!
- The "version" and "day/evening" markers on the form should not be filled in.

Question 1 Which statement about the representation of numbers is **false**?

a. All real numbers can be represented exactly in the so-called floating point representation.
b. Exponent and significand in the "float" and "double" formats are represented by a limited number of bits.
c. One bit in both the "float" and "double" format is used to represent the sign of the number.
d. A "float" is represented by 32 bits, while a "double" is represented by 64 bits.

Question 2 Method overloading means (which statement is the most correct one):

a. In 1 class you can define 2 or more methods with the same name.
b. In 1 class you can define 2 or more methods with the same name, but the names of the parameters need to be different.
c. In 1 class you can define 2 or more methods with the same name, but the types of the parameters need to be different.
d. In 1 class you can define 2 or more methods with the same name, but the parameter lists have to be different in terms of the number of parameters. If the number of parameters is equal, than the types of the parameters need to be different or the order of the types needs to be different.

Question 3 Given the following piece of code, what is printed on screen?

```
public static void main( String[] args ){
   Dog aDog = new Dog("Max");
   foo(aDog);

   if (aDog.getName().equals("Max")) {
      System.out.println("Max");
   } else if (aDog.getName().equals("Fifi")) {
      System.out.println("Fifi");
   }
}

public static void foo(Dog d) {
   d = new Dog("Fifi");
   if(d.getName().equals("Fifi")) {
         System.out.println("Fifi"); }
   if(d.getName().equals("Max")) {
         System.out.println("Max"); }
}
```

a. Fifi
   Fifi
b. Fifi
   Max
c. Max
   Max
d. Max
   Fifi

Question 4 Given the same code example as in Question 3, which statement is correct?

a. The parameter d of method foo is called the formal parameter.
b. The parameter d of method foo is called the actual parameter.
c. In the first line of the main method the default constructor is called.
d. In the first line of the method foo the default constructor is called.

Question 5 Which statement about encapsulation if **false?**

a. Is used to hide implementation from consumers of a class.
b. Is used as a protective measure against programmers creating or maintaining the class.
c. Eases changing the internals of a class, without worrying the consumer of the class.
d. Is particularly useless when using inheritance.

Question 6 How can you best describe how Java passes parameters?

a. Java is pass by value, this means that values of primitive types and memory addresses of reference types are copied.
b. Java is pass by value, this means that memory addresses of both primitive and reference types are copied.
c. Java is pass by reference, this means that values of primitive types and memory addresses of reference types are copied.
d. Java is pass by reference, this means that memory addresses of both primitive and reference types are copied.

Question 7 Consider a class Point with 2 attributes x and y, both of type int. Now consider this equals method:

```
public boolean equals(Point other){
        if (other instanceof Point){
                Point that = (Point)other;
                return (this.x == that.x) && (this.y == that.y);
        }
        return false;
}
```

In a test method, the following 3 lines of code are written:

```
Point a = new Point(3,4);
Point b = new Point(3,4);
assertEquals(a, b);
```

Given that these snippets of code are placed in their right context, this code:

a. Will not compile.
b. Will lead to a runtime error.
c. Leads to a successful test. → "pass"
d. Leads to a failing test → "fail"

Question 8 Java allows to (which answer is correct and most complete):

a. Inherit from multiple classes.
b. Inherit from multiple classes and multiple interfaces.
c. Inherit from a single class.
d. Inherit from a single class and multiple interfaces.

Question 9 Given a MouseListener, we can say:

a. MouseListener is an interface; if it contains > 1 method, there is an abstract MouseAdapter class
b. MouseListener is an interface; an abstract MouseAdapter class might exist.
c. MouseListener is a class; if it contains > 1 method, there is a MouseAdapter interface
d. MouseListener is a class; a MouseAdapter interface might exist.

Question 10 The difference between an abstract class and an interface

a. They are exactly the same thing, it is just another name.
b. An abstract class can have attributes, an implementation for (some) methods and abstract methods to define a contract. An interface can not have attributes, but it can have an implementation for (some) methods and abstract methods to define a contract.
c. An abstract class can have attributes, it cannot have an implementation for (some) methods and all of its methods should be abstract. An interface can not have attributes, it can not have an implementation for methods, it simply defines methods as a contract.
d. An abstract class can have attributes, an implementation for (some) methods and abstract methods to define a contract. An interface can not have attributes, it can not have an implementation for methods. It declares a contract.

Question 11 The method "accept()" is (most correct/complete answer):

a. Part of the class Socket. It waits for an incoming request for communication.
b. Part of the class Socket. It waits for an incoming request for communication. Furthermore, this method "blocks" the program until there is an incoming request.
c. Part of the class ServerSocket. It waits for an incoming request for communication. Furthermore, this method "blocks" the thread until there is an incoming request.
d. Part of the class ServerSocket. It waits for an incoming request for communication. Furthermore, this method "blocks" the program until there is an incoming request.

Question 12 Given the following piece of code:

```
@Test  public void testConstructor() {
        Address a1 = new Address("Mekelweg", 4);
        Address a2 = new Address("Mekelweg", 5);
        assertNotEquals(a1, a2);
}
```

a. Executing this test gives us confidence that both the constructor and the equals method are implemented correctly.
b. Executing this test gives us confidence that the constructor is tested well, but not the equals method.
c. Executing this test gives us confidence that the equals method is tested wel, but not the constructor.
d. None of the above.

Question 13 Given the following piece of code

```
public class ThreadWorker implements Runnable{

        private Thread worker;

        public ThreadWorker() {
                worker = new Thread(this);
        }

        public void start() {
                System.out.println("A new thread writes this");
        }
}
```

And this main:

```
public static void main(String[] args) {
        ThreadWorker tw = new ThreadWorker();
        tw.start();
}
```

a. This code does not compile
b. This code compiles and writes "A new thread writes this"
c. This code compiles, starts a new thread and that new thread writes "A new thread writes this".
d. This code compiles but ends with an exception.

Question 14 What is true about quality?
a. Internal and external quality are basically the same.
b. Internal quality refers to structural properties of the source code and the test code, while external quality refers to how nicely the code looks in terms of whitespaces, indentation, …
c. Internal quality refers to properties of the source code and the test code, while external quality refers to the user perspective (crashes, exceptions, …)
d. None of the above

Question 15 Consider the Model View Controller (MVC) design pattern. If we reason about this pattern in the context of a spreadsheet (e.g. Excel), which situation describes a possible implementation with MVC best?

a. The grid view with all the cells that a user can manipulate is the model. Graphs can be considered views.
b. The grid view with all the cells that a user can manipulate is a view, as are graphs. The model is an internal representation.
c. The grid view with all the cells that a user can manipulate is the model. Graphs can be considered views. A controller could for example be attached to the model so that changes in the model are propagated to the views.
d. The grid view with all the cells that a user can manipulate is a view, as are graphs. The model is an internal representation. A controller could for example be attached to the grid view, so that if that view is manipulated by the user, the underlying model is updated.

Question 16 Deadlock is:

a. Two or more threads competing for the same resources.
b. **Two or more threads waiting on each other before they can proceed**.
c. Inheriting from two classes and being unable to decide which parent attribute to use
d. Having an error (i.e. irrecoverable situation) and not being able to rectify things like releasing of locks or terminating network connections.

Question 17 Which interpretation of the word "binding" is **false**:

a. Binding is not relevant for the Java language.
b. Static binding means that the compiler decides on which method definition to associate to a method call.
c. Dynamic binding means that the virtual machine decides on which method definition to associate to a method call.
d. You can influence the binding through casting.

Question 18 Declaring a method abstract means that:

a. This method cannot have a return value.
b. This method must be overriden in a derived class
c. This method cannot be overriden in a derived class.
d. This method can only be called by other methods from the same class.

Question 19 Given the following class and its instantiations:

```
public class Pair<A, B> {
        private A a;
        private B b;
}

Pair<String, Integer> p1 = new Pair<>("foo", 1);
Pair<Integer, String> p2 = new Pair<>(1, "foo");
```

What statement describes the situation best?

a. System.out.println((p1 instanceof Pair)); //true
System.out.println((p2 instanceof Pair)); //true
→ because generic arguments are compile time only
b. System.out.println((p1 instanceof Pair)); //true
System.out.println((p2 instanceof Pair)); //true
→ because generic arguments are runtime only
c. System.out.println((p1 instanceof Pair)); //false
System.out.println((p2 instanceof Pair)); //false
→ because generic arguments are compile time only

d. System.out.println((p1 instanceof Pair)); //false
System.out.println((p2 instanceof Pair)); //false
→ because generic arguments are runtime only

Question 20 If we have 2 logical values A and B and we want to combine them we can make use of a lazy operator. Given A <operator> B, which statement about a lazy operator is true?

a. & → this operator does not evaluate B if A is false
b. && → this operator does not evaluate B if A is false
c. | → this operator does not evaluate B if A is false
d. || → this operator does not evaluate B if A is false

Question 21 Consider the code in Appendix A. Given that code, what is the result of the following 3 lines of code?

```
Train fasttrain = new InternationalTrain(true, "Amsterdam", "Brussels");
InterCity beneluxtrain = fasttrain;
System.out.println(beneluxtrain.capacityStandingPlaces());
```

a. A compile-time error
b. A run-time error
c. No error, "0" is printed to the screen.
d. No error, "150" is printed to the screen.

Question 22 Consider the code in Appendix A. Given that code, what is the result of the following 3 lines of code?

```
InterCity ic = new InterCity("Den Haag", "Vlissingen");
Train train = ic;
System.out.println(train.capacityFirstClass());
```

a. A compile-time error
b. A run-time error
c. No error, "0" is printed to the screen.
d. No error, "50" is printed to the screen.

Question 23 Consider the code in Appendix A. Given that code, what is the result of the following 3 lines of code?

```
Train crowdedtrain = new FreightTrain(1600, "Rotterdam", "Utrecht");
StopTrain littletrain = (StopTrain)crowdedtrain;
System.out.println(littletrain.maxSpeed());
```

a. A compile-time error
b. A run-time error
c. No error, "100" is printed to the screen.
d. No error, "120" is printed to the screen.

Question 24 The "instanceof" operator:

a. Checks whether a number can be represented as a particular primitive type (e.g. short).
b. Checks whether a cast from 1 primitive type (e.g. long) to another (e.g. int) has been carried out correctly.
c. Checks whether a cast from 1 referentce type to another has been carried out
d. Checks whether the dynamic type of an object (1st operand) is of (sub)type (2nd operand)

Question 25 Which statement is **untrue**? If you want to write good tests, you should:

a. Test each method at least once
b. Write as much asserts as you can per test method
c. Write extra peek and poke methods
d. Write extra get and set methods

Appendix A

```
public abstract class Train
{

	protected boolean electric;
	protected boolean diesel;

	private String from;
	private String to;

	public Train(boolean electric, boolean diesel, String from,
String to)
	{
		this.electric = electric;
		this.diesel = diesel;
		this.from = from;
		this.to = to;
	}

	public abstract int maxSpeed();
}

public class FreightTrain extends Train {

	private int torque;

	public FreightTrain(int torque, String from, String to)
	{
		super(false, true, from, to);
		this.torque = torque;
	}

	public int maxSpeed()
	{
		return 120;
	}
}


public interface TravelClasses {
```

```
	public int capacityFirstClass();

	public int capacitySecondClass();
}

public class StopTrain extends Train implements TravelClasses{

	public StopTrain (String from, String to)
	{
		super(true, false, from, to);
	}

	public int capacityFirstClass() {
		return 25;
	}

	public int capacitySecondClass() {
		return 125;
	}

	public int capaciteitStaanplaatsen() {
		return 80;
	}

	public int maxSpeed()
	{
		return 100;
	}

}

public class Sprinter extends StopTrain{

	public Sprinter(String from, String to)
	{
		super(from, to);
		this.electric = true;
	}

	public int maxSpeed()
	{
		return 120;
	}
}


public class InterCity extends Train implements TravelClasses {

	protected boolean minibar;

	public InterCity(String from, String to)
	{
		super(true, false, from, to);
		minibar = false;
	}
```

```
	public int capacityFirstClass() {
		return 50;
	}

	public int capacitySecondClass () {
		return 220;
	}

	public int capacityStandingPlaces() {
		return 150;
	}

	public int maxSpeed()
	{
		return 200;
	}

}


public class InternationalTrain extends InterCity {

	public InternationalTrain(boolean international, String from,
String to)
	{
		super(from, to);
		minibar = true;
	}

	public int capacityStandingPlaces()
	{
		return 0;
	}
}
```