Universidad Carlos III de Madrid

Systems Programming
Groups 65-69-79 and 95

Leganés, February 21st, 2014 — Mid-Term Exam 1 (test)
Duration: 10 min — Score: 5 points over 10

Only one answer is correct for each question. Each correct answer adds $\frac{1}{4}$ points. Each wrong answer substracts $\frac{1}{12}$ points. Not answered questions do not add nor substract points.

- No books or notes are allowed. Also, mobile phones and any other electronic devices must be off. Failure to comply with any of these norms can be reason enough for immediate expulsion out of the exam.
- Mark the answer to each question with an "X" in the table below.
- If none or more than one option is marked, the question is considered as not answered (thus not adding or substracting points).
- Fill in your **personal details** before you start the test.

Model: A

Name:	Group:

Signature:

NIA:

	A	B	C	D		A	B	C	D
1					6				
2					7				
3					8				
4					9				
5					10				

1.- Select which of the following statements is *false*.

(a) If a class implements an interface, all classes that inherit from that class also implement the interface, although not explicitly declared.

(b) Methods of an interface can contain no code.

(c) *** One class can implement just one interface.

(d) An interface can declare constants as attributes.

2.- Select which of the following statements is *false*.

(a) The composition of classes allows to have an object of a different class as attribute.

(b) We say that Java does not support multiple inheritance because a Child class cannot inherit at the same time from a Parent class A and a Parent class B, and this inheritance hierarchy is what is called multiple inheritance.

(c) A class cannot extend from more than one class but can implement from several interfaces.

(d) *** We state that Java supports multiple inheritance because a Child class can inherit from a Parent class that itself inherits from a GrandParent class, and this inheritance hierarchy is called multiple inheritance.

3.- Given the following class declarations, select which one is incorrect.

```
public class Student extends Person {...}
public class Teacher extends Person {...}
public class ErasmusStudent extends Student {...}
```

(a) `Object o = new Teacher();`

(b) `Student s = new Student();`

(c) ***

`Teacher t = new Person();`

(d) `Person p = new ErasmusStudent();`

4.- Select which of the following staments is *correct*, based the following code fragment:

```
Object o = new String("Good afternoon!");
char c = o.charAt(3);
```

(a) This only works if the `Object` class is a subclass of the `String` class.

(b) *** This does not work because, though there is a `charAt` method in `String` class, it cannot be invoked on a reference to an Object.

(c) This does not work because an object of the `String` class cannot be assigned to a reference of the `Object` class.

(d) This works correctly and stores the character `'d'` into `c`.

5.- Select which sentence is needed to add a new button to the *JFrame* window referenced by *frame* variable.

(a) `frame.getContentPane().add(JButton)`

(b) ***

```
frame.getContentPane().add(new JButton())
```

(c) `getContentPane().add(new JButton())`

(d) `(new JButton()).getContentPane().add(frame)`

6.- Assume that we have a `Ship` abstract class with an abstract method called `public void navigate()` and two classes called `MerchantShip` and `ArmyShip` that inherit from the first one and implement the method. Given the following code, select which of the following statements is correct.

```
MerchantShip b1 = new MerchantShip("Ship-A");
ArmyShip b2 = new ArmyShip("Ship-B");
Ship[] fleet = {b1,b2};
for(int i=0; i<fleet.length; i++) {
  fleet[i].navigate();
}
```

(a) The code is incorrect and downcasting should be done to solve the problem.

(b) The code does not compile because we are invoking the `navigate` method on `Ship` objects, and the `navigate` method in `Ship` class is abstract.

(c) *** The code is correct and which of both implementations of `navigate` is going to be used is decided at runtime.

(d) The code is incorrect and to solve the problem a conditional should be included to guess the class of each object in the array and thus to know which implementation of `navigate` should be used.

7.- Select which package has to be imported to use `JButton` class in our code.

(a) `import java.awt.*`

(b) *** `import javax.swing.*`

(c) `import java.awt.event`

(d) `import java.graphics.*`

8.- Select the *false* answer.

(a) It is possible that there exists a method in both a Parent class and a Child class with the same name and same signature.

(b) *** Method overwriting or overriding is when there exist two methods with the same name and different signature (different parameters or output).

(c) when a constructor is overloaded, it can be called from another constructor in the same class using `this()` with the appropriate parameters.

(d) A constructor in a Child class can only call the constructor in the Parent class if it is done in the very first sentence of the constructor code.

9.- Select which parameter must be passed to *addActionListener* in the following code so that the button listens to the events fired when a user clicks on it.

```
public class MyButton extends JButton implements ActionListener {
    private int counter;
    public MyButton() {
        super("No clicks yet.");
        counter = 0;
        addActionListener(.........);
    }
    public void actionPerformed(ActionEvent e) {
        counter++;
        setText("I have been clicked " + counter + " times.");
    }
}
```

(a) `JButton`

(b) `***`

`this`

(c) `actionPerformed()`

(d) `new ActionListener()`

10.- There is a `Person[] data` array that stores objects of the `Fireman` and `Policeman` classes, where both of them inherit from `Person` class. If an object of the `Fireman` class is inserted into the first position of the array and we want to invoke its method `putFireOut()`, select which of the following choices is correct.

(a) putFireOut(data[0])

(b) data.putFireOut(0)

(c) *** ((Fireman)data[0]).putFireOut()

(d) data[0].putFireOut()