### **Week 5 - EXAM**

### Abstract classes

0.0/2.0 points (graded)

Given the following code where two classes are defined:

**public abstract class Figure {**

**public String name;**

**public Figure (String n){**

**name=n;**

**}**

**public String toString(){**

**return name;**

**}**

**public abstract double calculateArea();**

**public abstract double calculatePerimeter();**

**}**

**public class Circle extends Figure{**

**private double radius;**

**private final double PI=3.1415;**

**public Circle (double r, String name){**

**super(name);**

**radius=r;**

**}**

**public String toString(){**

**return "I'm a circle of radius "+radius;**

**}**

**public double calculateArea(){**

**return PI\*radius\*radius;**

**}**

**public double getPerimeter(){**

**return 2\*PI\*radius;**

**}**

**}**

Select which sentences are correct

**Be sure that you mark all the correct answers (*Wrong answers subtract points*).**

1. This code will compile, but will give an error during execution if the method calculatePerimeter() is called from another class
2. **This code will not compile (Answer)**
3. **The class Figure cannot be instantiated, as it is an abstract class (Answer)**
4. Circle does not need to implement the method calculatePerimeter() as it already has a getPerimeter() method
5. The class Circle is a subclass of Figure, and thus all the methods in Figure are automatically implemented in Circle
6. **The class Circle cannot be instantiated, as it inherits from an abstract class (Answer)**

**Explanation:**

The code will not compile, because the method calculatePerimeter() defined in the class Figure is not implemented in the class Circle. Moreover, abstract classes cannot be instantiated.

Imagine that the class Secretary extends the abstract class Employee. Select which statements are correct to create an instance of the Secretary class.

**Be sure that you mark all the correct answers (*Wrong answers subtract points*).**

1. Secretary secretary=new Employee();
2. **Secretary secretary=new Secretary(); (Answer)**
3. **Employee secretary=new Secretary(); (Answer)**
4. Employee secretary=(Secretary)(new Employee());

**Explanation:**

The class Employee is abstract, and thus cannot be instantiated

### 2. Class Integer: Max value and Min value

0.0/1.0 point (graded)

Given the following code:

**public class HelloInteger{**

**public static void main(String []args){**

**System.out.println(Integer.MAX\_VALUE + Integer.MIN\_VALUE);**

**}**

**}**

Enter the number printed on screen by this program: **-1 (Answer)**

### 3. The correct statement

0.0/2.0 points (graded)

For each of the following sets of statements, choose the correct one.

1. An abstract class does not contain non-abstract methods (except for static methods).
2. Both abstract classes and interfaces require a constructor method to be implemented.
3. **An interface does not contain non-abstract methods (except for static methods (Answer)**

2.

1. Interfaces are not allowed in Java versions prior before Java 8.0.
2. It is possible to instantiate an interface if it has been extended by other classes.
3. **An interface may have attributes, but they must be static and final (constants). (Answer)**

3.

1. For a given abstract class, all its methods must be overridden in the classes that extend this abstract class.
2. **For a given abstract class, all its abstract methods must be overridden in the classes that extend this abstract class. (Answer)**
3. An abstract method can be partially implemented in the abstract class, but it must be fully implemented in the classes extending this abstract class.

4.

1. **One difference between interfaces and abstract classes is that the latter can implement non-abstract methods (Answer)**
2. One difference between interfaces and abstract classes is that the formers only contain final parameters
3. There are no differences between interfaces and abstract classes

### 4. Drawing the UML Class Diagram

0.0/3.0 points (graded)

Given the two UML blocks for classes shown in the Figure, select which elements should be included in each part marked with a circled number. You must infer the required information from the source code of the class Elephant, which is:

**public class Elephant extends Animal{**

**public String origin; // ASIA or AFRICA**

**public void bathe (){**

**System.out.println (super.owner+" must bathe the elephant from "+origin);**

**}**

**public boolean ride (){**

**if (super.age < 5){**

**System.out.println ("The elephant is to young to ride");**

**return false;**

**}else if (super.isSleeping()){**

**System.out.println ("The elephant is sleeping now");**

**return false;**

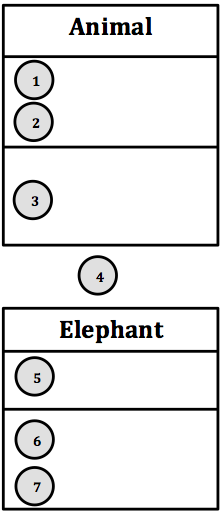
**}else{**

**return true;**

**}**

**}**

**}**

****

1.

1. + void bathe()
2. + boolean ride()
3. **+ String owner (Answer)**
4. + String origin
5. + void feed(Food[]food)

2.

1. + void bathe()
2. **+ int age (Answer)**
3. + boolean isSleeping()
4. + String origin
5. + void feed(Food[]food)

3.

1. + void bathe()
2. + int age
3. **+ boolean isSleeping() (Answer)**
4. + boolean ride()
5. + String origin

4.

1. ****
2. **(Answer)**

5.

1. + void bathe()
2. + String age
3. + boolean isSleeping()
4. + String owner
5. **+ String origin (Answer)**

6.

1. + boolean isSleeping()
2. **+ boolean ride()**
3. + String owner
4. + String origin
5. + void feed(Food[]food)

7.

1. **+ void bathe()**
2. + String age
3. + boolean isSleeping()
4. + String owner
5. + String origin
6. + void feed(Food[]food)

**Explanation:**

****

### 5. String methods

**0.0/3.0 points (graded)**

Given the following code:

**String a=new String ("abcdefg");**

**String b=new String ("ABCDEFG");**

**String c=new String ("AbcdefG");**

**String d=new String ("cde");**

What are the results printed on screen by the following statements:

**System.out.println (a.equalsIgnoreCase(b));**

1. false
2. **true (Answer)**

**Explanation:**

The method equalsIgnoreCase ignores case considerations, and thus a and b look the same sequence

**System.out.println (c.substring(2,4));**

1. cde
2. **cd (Answer)**
3. def
4. de

**Explanation:**

The method substring returns the sequence that begins at the specified first index ('c' in this case) and extends to the character at the index in the second argument - 1 ('d' in this case)

**System.out.println (a.equals(b));**

1. **false (Answer)**
2. true

**Explanation:**

a and b are not the same String as they do not have the same characters

### 6. Understanding the code

**0.0/3.0 points (graded)**

**Given the following program written in Java:**

**package myUtils.system;**

**import java.io.\*;**

**class UtilitiesSystem{**

**public static void main (String args[]){**

**foo();**

**}**

**public static void foo (){**

**File pwd=new File(System.getProperty("user.dir"));**

**if (pwd.isDirectory()){**

**File [] bar=pwd.listFiles();**

**for (int i=0;i<bar.length;i++){**

**System.out.println (bar[i].getName());**

**}**

**}else{**

**System.out.println ("Error. The argument is not a directory");**

**}**

**}**

**}**

Select the correct sentences.

**Be sure that you mark all the correct answers (*Wrong answers subtract points*).**

1. **Any external class can access the method foo() from class UtilitiesSystem, after importing the package "myUtils.system". (Answer)**
2. The program will not compile, as the class System is not imported.
3. **All classes from the package "java.io" are available in this program (Answer)**
4. Any external class can access the method foo() from class UtilitiesSystem, after importing the package "myUtils".
5. **The method foo() prints the directories of the system. (Answer)**
6. **The method foo() prints all the files and directories of the directory from which the program is executed. (Answer)**
7. The program will not compile, as the class File is not imported.
8. The message "Error. The argument is not a directory" will be printed in case the variable "user.dir" is not initialized.
9. **The message "Error. The argument is not a directory" is never printed. (Answer)**

**Explanation:**

1. The program will compile, as the class File is imported, same as every other classes from the package "java.io". Besides, the class System belongs to the package java.lang, and thus it is imported by default.

2. The method foo is public, and the class belongs to the package "myUtils.system". Thus, it is required to import this package in order to access the method foo() (importing the package "myUtils" is not enough.

3. The method foo() prints all the files and directories of the directory from which the program is executed. This directory is obtained by getting the "user.dir" property of the system, and looping through the array of Files contained within this directory. Check the API documentation for more details.

4. The message "Error. The argument is not a directory" is never printed. This is because the property "user.dir" always returns a directory path, and thus the condition within the "if" is always true.

### 7. The area of a circle

0.0/3.0 points (graded)

Suppose that you want to create a method that returns the area of a circle, whose radius is given by parameter. To avoid negative values for the parameter received, this method internally transforms it into positive by using the absolute value. Complete the following method **circleArea** using **methods and attributes from the class Math**. Indicate the correct attribute or method of the class Math that must replace the hyphens in lines 2, 3 and 4.

**Line 1: public double circleArea (double radius){**

**Line 2: double first=Math.--;**

**Line 3: double second=Math.---(radius);**

**Line 4: double third=Math.---(second,2);**

**Line 5: return first\*third;**

**Line 6: }**

Line 2: **PI (answer)**

Line 3: **abs (answer)**

Line 4: **pow (answer)**

**Explanation:**

The formula in Java for the area of a circle is Math.PI\*Math.pow(Math.abs(radius),2);

### 8. Finding the appropriate method

0.0/3.0 points (graded)

Use the Standard Java Library documentation to find information of classes and methods that perform the following operations. Indicate the exact name of the package, the name of the class and the name of the method (without parenthesis nor arguments).

For example, if the question were to find the package, class and method that calculates the square root of a number, then the correct answer would be:

|  |  |  |  |  |
| --- | --- | --- | --- | --- |
| **Package** |  | **Class** |  | **Method** |
| java.lang |  | Math |  | sqrt |

\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_

**1- Determine the current system properties**:

|  |  |  |
| --- | --- | --- |
| **Package** | **Class** | **Method** |
| java.lang | System | getProperties |

**Explanation:**

The method getProperties() from class System provides an instance of class Properties, which allows you to check properties such as the current system time or the operating system version.

**2 - Get the default time zone for the host where the program is running**:

|  |  |  |
| --- | --- | --- |
| **Package** | **Class** | **Method** |
| java.util | TimeZone | getDefault |

**Explanation:**

When writing, for example, network applications, it is very important to synchronize the systems being connected. Thus, as programmers, you should be aware of the time zone where the programs are running. The method getDefault() from class TimeZone provides you with such information.

**3 - Check whether a file is a directory**:

|  |  |  |
| --- | --- | --- |
| **Package** | **Class** | **Method** |
| java.io or java.nio | File | isDirectory |

**Explanation:**

The class File is used to represent disk files. The method isDirectory() allows you to know whether the represented file is actually a file or a directory. It returns a boolean (True if it is a directory, False otherwise).

Since Java version 1.7, a new package java.nio.file was added, which contains the class File. This class consists exclusively of static methods that operate on files, directories, or other types of files.