Introductory Programming Midterm-resit

Multiple-choice questions for GP21

```java
public class Car {
  private int speed;
  private int accel;
  private static int count=0;

  public Car(){
    speed=10;
    accel=2;
  }
  public Car(int speed, int accel){
    speed=speed;
    this.accel=accel;
    count++;
  }
  public int getCount(){
    return count;
  }
}
```

Listing 1: Car example

1. **Java Classes**

   Consider the code in Listing 1. Given that we create an object by `Car myCar = new Car(20,3);` what are the values of the fields of `myCar`? (Points: 1)

   (a) `speed = 20, accel = 2`
   (b) `speed = 20, accel = 3`
   (c) `speed = 0, accel = 3`
   (d) `speed = 0, accel = 2`
   (e) `speed = 10, accel = 3`

2. **Java Classes**

Consider the code in Listing 1. After running:

```
Car car1=new Car(30,4); Car car2=new Car(); Car car3 = new Car(20, 5);
```

what is the return value of `car1.getCount()`? (Points: 2)

(a) `0`
(b) `1`
(c) `2`
(d) `3`

3. **Operators**

   Assume `x` is a variable of type `int`. Which of the following expressions evaluate to the same result as `x % 15 == 0`? (Points: 2)

   (a) `x % 5 == 0 ? false : x % 3 == 0`
   (b) `x % 15 == 0 ? false : true`
   (c) `x % 5 == 0 ? x % 3 == 0 : false`
   (d) `x % 3 == 0 ? x % 5 == 0 : true`

4. **Increments**

   Consider two variables defined as:
   `int x = 5;` and `String s=((x++) + " " + (++x));` What is the result of printing `s`? (Points: 1)

   (a) `5 6`
   (b) `5 7`
   (c) `6 6`
   (d) `6 7`
   (e) none of the above

5. **Collections**

   Consider the code `for(String s : collection) { System.out.print(s); }`. Which of following can be the type of `collection`? (Points: 2)

   (a) `String[]`
   (b) `HashMap<String,Integer>`
   (c) `HashSet<List<String>>`
   (d) `ArrayList<String>`

6. **Interfaces**

   Consider three classes `A`, `B`, and `C` and interfaces `I` and `J`. Which of the following class or interface declarations are *not* valid? (Points: 2)

   (a) `public interface I extends J`
   (b) `public class A extends B,C implements I`
   (c) `public interface I extends B implements J`
   (d) `public class A implements I,J`

```java
public class A {
  public A() { System.out.print("A"); }
}

public class B extends A {
  private Student student;
  public B(Student student) {
     System.out.print("B");
     this.student=student;
  }
}
```

Listing 2: Inheritence

7. **Inheritence**

   Consider the code in Listing 2. After running `B b = new B(student);`, what is printed? (Points: 2)

   (a) An error occurs as the constructor of A should be called in the constructor of B
   (b) `A`
   (c) `B`
   (d) `BA`
   (e) `AB`

8. **Type inference**

   Consider a method `public static void method(Set<Integer> a)` in Class A. Assuming that we run `A.method(new HashSet<Integer>())`, what are the static and dynamic types of the parameter `a`? (Points: 2)

   (a) static type `HashSet<Integer>`, dynamic type `Set<Integer>`
   (b) static type `HashSet<>`, dynamic type `HashSet<Integer>`
   (c) static type `Set<Integer>`, dynamic type `HashSet<>`
   (d) static type `Set<Integer>`, dynamic type `HashSet<Integer>`

9. **Collections**

   Suppose `myMap` is a variable of type `HashMap<String,Integer>`. What are the types of `x` and `j` in the following line: `var x = myMap.get(j);`? (Points: 1)

   (a) `x` has type `int`, `j` has type `Integer`
   (b) `x` has type `Integer`, `j` has type `Integer`
   (c) `x` has type `Integer`, `j` has type `String`
   (d) `x` has type `String`, `j` has type `int`
   (e) `x` has type `int`, `j` has type `String`

```java
public class MyClass {
  public int myMethod() {
    try { throw new Exception("New exception!"); }
    catch (Exception e) { return 8; }
    finally { System.out.print("_finally_"); }
  }
}
```

Listing 3: Exceptions

```java
public class  MyClass{
  public boolean myMethod(boolean a, boolean b) {
      return !(a && b);
  }
}
```

Listing 4: Testing

10. **Operators**

    Considering the line `int a=4;`. What is the result of printing the string "1"+a+1+"2"? (Points: 1)

    (a) 152
    (b) 1412
    (c) 512
    (d) 53

11. **Exceptions**

    Consider the code in Listing 3. What is the result of running
    `MyClass m=new MyClass(); System.out.print(m.myMethod());` (Points: 2)

    (a) `_finally_`
    (b) `_finally_8`
    (c) `8`
    (d) `8_finally_`
    (e) The code cannot run since the `try` block does not return a result of type `int`.

12. **Testing**

    Consider the code in Listing 4 and the line `MyClass m = new MyClass();` Which one of the following assertions are suitable to test `myMethod`? (Points: 2)

    (a) `assertEquals(m.myMethod(true,false),false)`
    (b) `assertTrue(m.myMethod(true,true))`
    (c) `assertTrue(true,m.myMethod(false,true))`
    (d) `assertEquals(false,m.myMethod(true,true))`

```
public class MyClass {
  public static int a = 3;
  public int[] arr = {4,1,0,3,8};
}

public class MyClass2 {
  public void myMethod() {
    MyClass x,y;
    x = new MyClass(); y = new MyClass();
    for(int i = 0; i < y.arr.length; i++){
      if(i == y.arr[i]) {
        x.arr[i]++;
        x.a++;
      }
    }
    //(1)
  }
}
```

Listing 5: Array assignment

13. **Assignment**

Consider the code in Listing 5. Which of the following expressions are true at (1) (Points: 2)

(a) `y.a == 4`
(b) `x.a == 5`
(c) `x.arr[3] == 4`
(d) `y.arr[1] == 3`
(e) `x.a == y.a`

14. **Object dereferencing**

Consider the code `String s = null; int length = s.length();`. What is the value of the variable `length` after the code has executed? (Points: 1)

(a) `0`
(b) `1`
(c) The exception `IllegalArgumentException` is thrown.
(d) The exception `NullPointerException` is thrown.
(e) The code cannot be run as possible checked exceptions are not propagated.

Programming Problem

In the following you will find the description of the programming problem. In this part, you need to implement a service robot that transfers boxes from an original position to a specified destination for each box. The boxes have labels and the priority of transferring a box depends on its label. Implement the Robot and Box classes as described below. Please note that it is important that the naming of your project, classes, and signature of methods exactly follow the description. All fields must be private and all methods must be public. (Points 77)

1. Define two classes `Robot` and `Box`.

2. The class `Box` has three fields `position` and `destination` of type `int` and `label` of type `String`. The fields `destination` and `label` are initialised in the constructor of the class with values of the two parameters of the constructor. All boxes have 0 as initial value for their `position`.

3. The class `Robot` has one field named `name` of type `String`. Furthermore, a field `transferList` of type `List<Box>` and another field named `labelPriorities` of type `Map<String,Integer>`. The constructor of the class has only one parameter of type `String` that is used for initialisation of the field `name`.

4. In class `Box`, implement three methods `getPosition`, `getDestination` and `getLabel` which have no parameter and return, respectively, the position, destination and the label of a box.

5. In class `Box`, implement a method `forward` that receives no parameter and increases the position of box by 1.

6. In class `Robot`, implement a method `addToTransferList` which adds boxes to the list to transfer. This method receives an object of type `Box` as its only parameter and adds it to `transferList`. The method returns `true` if the addition is successful and `false` otherwise.

7. In class `Box`, override the `toString` method such that for a box, with position `p` and label `l`, the method returns a String with the following format:
   `"position: p, label: l"`
   Note that the method does not have any parameters. (Points: 6)

8. In class `Robot`, implement a method `printTransferList` that prints the information about boxes in the `transferList` per line. For example, for a `transferList` that contains two boxes, `b1` with position `p1` and label `l1`, and box `b2` with position `p2` and label `l2`; the following will be printed after running the method:
   `"position: p1, label: l1"`
   `"position: p2, label: l2"`

   This method does not have any parameter and does not return anything. (Hint: use the `toString` method in class `Box` as a helper method.)

9. In the class `Robot`, implement a `main` method. In the main method, create an object of type `Robot` and two objects of type `Box`. Add the two objects to the `transferList` using the methods above. Print the list of boxes to be transferred using the `printTransferList` method.

10. In class `Robot`, implement a method `moveForward`, which has two parameters called `box` and `steps` of types `Box` and `int`. The robot moves the `box` forward and `steps` represents how much the position of the `box` increases. However, a box may not be moved further than its `destination` and will remain in place once it reaches it. (Hint: this method uses method `forward` from the `Box` class as a helper method.)

11. In class `Robot`, implement a method `setLabelPriority` which receives a parameter called `label` of type `String` and another parameter called `priority` of type `int` that represents the priority of that label. If `String` is not `null`, then the method maps the key `label` to `priority` in `labelPriorities`.

12. In class `Robot`, implement a method `getBoxPriority` that returns the priority of a box (an `int`) based on the label of the box. If the label of the box is not a key in `labelPriorities` the method throws an `IllegalArgumentException` exception with the message `"Invalid label!"`.

13. In class `Robot`, implement a method `selectNextBox`, which selects a box from `transferList` with highest priority to move. The method returns the selected box. Once the box with highest priority is selected, it is removed from the `transferList`. If there are several boxes with highest priority, only one will be selected, and it does not matter which one you pick. If the `transferList` is empty, the method returns `null`. (Hint: the method uses `getBoxPriority` as a helper method.)

14. In class `Robot`, implement a method `removeFromTransferList`, which receives a parameter `labelRem` of type `String`. The method removes all boxes that have a label with value of `labelRem` from the `transferList`.