Introductory Programming Midterm exam

Multiple-choice questions for GP21

```java
public class Card {
  private String message;
  private int numHearts;
  // (1) definition of Card constructor

  public static void printBirthdayMessage() {
    System.out.println("Happy Birthday!!!");
  }

  public void updateMessage(int numHearts, String msg) {
    this.message = msg; this.numHearts = numHearts;
  }
  ...
}
public class Main {
  public static void main(String[] args) {
    //(2) construct a Card object called myCard
    //(3) print the happy birthday message
    //(4) update myCard
  }
}
```

Listing 1: Card example

1. **Java Classes**

   Consider the code in Listing 1. How do you start writing a constructor for the class `Card` at (1)? (Points: 1)

   (a) `public Card {...`
   (b) `public void Card(){...`
   (c) `public Card(void){...`
   (d) `public Card(){...`

2. **Java Classes**

   Consider the code in Listing 1. How do you construct an object of class `Card`, named `myCard`, at (2)? (Points: 1)

   (a) `Card mycard;`
   (b) `Card myCard = Card();`
   (c) `Card myCard = new(Card);`
   (d) `myCard.Card();`
   (e) `Card myCard = new Card();`

3. **Java Classes**

   Consider the code in Listing 1. How do you print the happy birthday message at (3)? (Points: 2)

   (a) `myCard.printBirthdayMessage();`
   (b) `printBirthdayMessage();`
   (c) `Card.printBirthdayMessage();`
   (d) `printBirthdayMessage(myCard);`

4. **Java Classes**

   Consider the code in Listing 1. How do you call the updateMessage method at (4)? (Points: 1)

   (a) `Card.myCard.updateMessage(2, "Happy Birthday!")`
   (b) `Card.updateMessage(2, "Happy Birthday!")`
   (c) `updateMessage(myCard(2, "Happy Birthday!"));`
   (d) `updateMessage(2, "Happy Birthday!");`
   (e) `myCard.updateMessage(2, "Happy Birthday!");`

5. **Java Classes**

   Which of the following statements are correct? (Points: 2)

   (a) Constructors are always declared as private
   (b) The constructors can appear anywhere in the class where it is legal to declare a method
   (c) The return type of constructors can be `void`.
   (d) If `A extends B` then the constructor of `A` should always have parameters

6. **Operators**

   Which of the following expressions evaluate to the same result as `(a>b) && !(b<c)`? (Points: 2)

   (a) `(a>b) ? !(b<c) : true`
   (b) `!(b<c) ? (a<b) : true`
   (c) `(a>b) ? !(b<c) : false`
   (d) `(b<c) ? (a<b) : false`

```
String[] result =
   "this is a word in this sentence in this".split(" ");
Set<String> mySet=new HashSet<String>();
for(var v : result) { mySet.add(v); }
for(var s : mySet) {
  if(!s.equals("this")) System.out.print("1");
}
```

Listing 2: Printing ones

7. **Collections**

   Consider the code in Listing 2. What are the possible results of running the code? (Points: 3)

   (a) 1
   (b) nothing is printed
   (c) 11111
   (d) 111111
   (e) 11111111

8. **Interfaces**

   Which of the following class headers correctly defines a subclass of `S` that is guaranteed to have all functions from the interface `I`? (Points: 1)

   (a) `public class myClass extends I implements S`
   (b) `public class myClass extends S throws I`
   (c) `public class myClass extends I, S`
   (d) `public class myClass extends S implements I`

9. **Methods**

   Java `String` objects have the method `public String substring(int beginIndex , int endIndex)` which returns a `String` object containing a substring with start position `beginIndex` and end position `endIndex` (not including the character at `endIndex`).
   Suppose `String x = "orange,apple,banana"`, which of the following expressions returns the string `"ppl"`? (Points: 2)

   (a) `x.substring(1,9).substring(1,3);`
   (b) `x.substring(5,10).substring(0,2);`
   (c) `x.substring(5,12).substring(2,5);`
   (d) `x.substring(4,11).substring(4,7);`
   (e) `x.substring(0,6).substring(3,6);`

10. **Type inference**

Suppose `myMap` is of type `HashMap<Integer, List<String>>`. What is the type of variable `s` in the loop `for (var s : myMap.keySet()){...}`? (Points: 2)

(a) `int`
(b) `Integer`
(c) `String`
(d) `List<String>`

11. **Collections**

Suppose `myMap` is a field of type `TreeMap<Student, List<String>>`. In your code you find the following variable declaration: `int x = myMethod(myMap.get(param));`. What are possible signature(s) of myMethod? (Points: 2)

(a) `int myMethod(Entry<Student, List<String>> s)`
(b) `public int myMethod(Student s)`
(c) `public myMethod(String s)`
(d) `int myMethod(List<String> s)`

12. **Collections**

Have `B` be a subclass of `C` and consider `C v = myMethod();`, where the return type of `myMethod` is `B`. What is the static type and the dynamic type of `v`? (Points: 3)

(a) static type C, dynamic type C
(b) static type C, dynamic type B
(c) static type B, dynamic type C
(d) static type B, dynamic type B

13. **Exceptions**

For two `int` variables `i` and `j` and an array `arr`, which of following types of exception can be thrown when evaluating the expression `i/arr[j]++`? (Points: 2)

(a) `IndexOutOfBoundsException`
(b) `IllegalArgumentException`
(c) `IOException`
(d) `ArithmeticException`

14. **Assertions**

Which statements about Java `assert` statements are true? (Points: 2)

(a) Assertions should have side effects
(b) An assertion that is not true throws an exception
(c) If an asserted statement is false an error is thrown
(d) The Java runtime machine has a flag related to assertions

```java
public class A extends B {
  public String toString() {
    return "ID" + super.toString();
  }
}
public Class MyClass {
  public String myMethod() {
    A a = new A();
    B b = new B();
    B c = a;
    return // (1)
  }
}
```

Listing 3: Overriding methods

15. **Method overriding**

Consider the code in Listing 3. What is true about the return value of `myMethod`? (Points: 3)

(a) Calling `a.toString()` at (1) always returns a string starting with "ID"
(b) Calling `b.toString()` at (1) always returns a string starting with "ID"
(c) Calling `c.toString()` at (1) always returns a string starting with "ID"

16. **Packages and import**

What is an effect of having `import java.util.HashSet;` at the top of your Java file? (Points: 1)

(a) It allows you to use the `new Set<>()` constructor
(b) It allows you to use the `new HashSet<>()` constructor
(c) A `HashSet` object is created when the program starts
(d) It turns all sets into `HashSet` objects

17. **Inheritance**

Consider the classes: `public abstract class C {...}` and `public final class D extends C {...}`.
Which of the following statements are true? (Points: 3)

(a) It is possible to create an instance of class `C`.
(b) It is possible to create an instance of class `D`.
(c) It is possible to create another subclass `E` of class `C`.
(d) It is possible to create a subclass `F` of class `D`.
(e) It is possible to create another superclass `G` of class `D`.

```java
public Boolean myMethod(int a, int b) {
  return (a >= b && a % 3 == 0)
}
```

Listing 4: Testing

```java
public class myClass {
  public static int a = 2; public int[] b = {1,5,6,9,8};
}
public class myClass2 {
  public void myMethod() {
    myClass x,y;
    x = new myClass(); y = new myClass();
    x.a += 1; x.b[x.a] += 1; y.a += 1; y.b[y.a] += 1;
    // (1)
  }
}
```

Listing 5: Array assignment

18. **Testing**

Consider the code in Listing 4. Which one of the following assertions are suitable to test `myMethod`? (Points: 2)

(a) `assertEquals(true, myMethod(8,3))`
(b) `assertTrue(myMethod(6,10))`
(c) `assertEquals(false,myMethod(7,5))`
(d) `assertEquals(myMethod(6,10),true)`

19. **Assignment**

Consider the code in Listing 5. Which of the following expressions are true at (1) (Points: 3)

(a) `x.a == 4`
(b) `y.a == 3`
(c) `x.b[3] == 9`
(d) `y.b[4] == 9`

```
1 public class myClass {
2   private int distance; private String name;
3   public void myMethod(int distance, String name) {
4     distance=distance; this.name=name;
5   }
6 }
```

Listing 6: Field assignment

20. **Assignment**

Consider the code in Listing 6. What is the effect of running `myMethod`? (Points 2)

(a) Fields `name` and `distance` have both changed to the values of method parameters `name` and `distance`

(b) The value of the field `name` has changed to the value of the method parameter `name`

(c) The value of the field `distance` has changed to the value of the method parameter `distance`

(d) none of the above answers is always correct

In the following you will find the description of a system for managing a laboratory. A laboratory has a set of available equipment and students can sign up to and sign off from a laboratory. Additionally they can book equipment to work with. Implement the classes for laboratory and students 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. For methods where the signature is not fully described, implement a method with an appropriate signature. (Points 60)

1. Define two classes `Lab` and `Student`.

2. The `Lab` class has two fields `name` and `capacity` of types `String` and `int`, respectively. These two fields are initialised in the constructor of the class with values of the two parameters of the constructor.

3. The `Student` class has a field `name` of type `String` and a field `age` of type `int`. The constructor of the class initialises the field `name` with the value of its parameter.

4. The `Student` class also has two methods `setAge` and `getAge` which are used, respectively, to set the value of age to different values and to return the value of age.

5. Define a field named `students` of type `List<Student>` in the `Lab` class.

6. In the `Lab` class create the following two methods that take a `Student` as an argument and return `true` if the operation succeeds and `false` if it fails:

   - a method `register` which registers a student (adds it to `students`) if the lab capacity is not full and if the student is not already registered.
   - a method `remove` that removes a student from the list if the student is registered.

7. Define two methods, `signUp` and `signOff` in the `Student` class which are responsible for signing up the student to a lab and signing off the student from a lab, using the `register` and `remove` methods from `Lab` as helper methods. These new methods should take a `Lab` as an argument and return `true` if the operation was successful and `false` otherwise.

8. A lab contains available equipment. Define a field in the `Lab` class which is named `availableEquipment` of type `Set<String>`. Each member in this set is the name of a piece of equipment that is available in the laboratory, for example "barometer".

9. Implement a method `addEquipment` in the class `Lab` such that it receives the names of equipment from the standard input and adds them to the set `availableEquipment`. In this method you need to use a Scanner to get the input from the standard input. The first line includes a whole number representing the number of pieces of equipment that are going to be added. Assuming that the value n is in the first line; the next n lines in the input each include a name of a piece of equipment. For example reading the following lines from the standard input

```
3
"hammer"
"spanner"
"mallet"
"this row will not be read because the number was three"
```

should add the strings `"hammer"`, `"spanner"`, and `"mallet"` to `availableEqipment`.

10. Define a field `studentEquipment` in the `Lab` class of type `Map<Student, Set<String >>` representing all equipment that has been booked by a student. Each entry in `studentEquipment` contains a `Student` object mapped to a set of equipment names i.e., set of strings, that the student has borrowed.

11. In the class Lab, implement a method with signature `boolean bookEquipment( Student student, String equipmentName)` that adds the single piece of equipment called `equipmentName` to the equipment booked by `student` and removes `equipmentName` from the available equipment of the lab. If the lab does not contain `equipmentName` throw an excption of the type `IllegalArgumentException` with the message `"Equipment not available!"`.

12. In class `Lab` implement a method `void returnEquipment(Student student)` which adds all the equipment booked by the `student` to the set `availableEquipment` and furthermore updates the map `studentEquipment` by removing the entry with key `student`.

13. Define a class `GradStudent` as a subclass of the Student class.

14. Each graduate student can at most be signed up to three labs at any one time. To implement this restriction, add necessary fields and override the `signUp` and `signOff` methods by making the necessary changes. Make sure that any relevant state of `GradStudent` only updates when you successfully register or deregister from a lab.