NAME : Bardia
LAST.NAME : Ardakanian
Studentt number : 9831072

Q1:
A **framework is** a collection of program that you can use to develop your own application.
It **is** built on top of a **programming language**. **Framework is** a set of pre-written code libraries designed to be used by developers. A **programming language is** a specified method of communication **between** the **programmer** and computer.

Spring

With its concept of Dependency Injection and aspect-oriented programming features, Spring took the development world by storm. It is an open-source framework used for Enterprise applications.

With Spring, developers can create loosely coupled modules where-in dependencies are handled by the framework rather than depending on the libraries in the code.

Spring framework is exhaustive and covers a lot of features including security and configuration, which are easy to learn. Further, since it is the most popular web framework, you can find a lot of documentation and an active community.

With everything configured, your code will be clean and easy to comprehend.

Dropwizard

Another Java framework true to its name – wizard. This light-weight framework lets you complete your application very fast because of it's out of the box support for advanced configurations, logging, application metrics and much more. You can create RESTful web applications that give high performance, are stable and reliable.

Dropwizard is especially magical because it brings together a host of libraries like Jetty, Guava, Jersey, Jackson, and Metrics amongst many others from the Java ecosystem into one framework and gives you a light-weight and lean application.

Q2:

a)
Structured programming is a programming paradigm aimed at improving the clarity, quality, and development time of a computer program by making extensive use of the structured control flow constructs of selection (if/then/else) and repetition (while and for), block structures, and subroutines.

Object-oriented programming (**OOP**) is a programming paradigm based on the concept of "objects", which can contain data, in the form of fields (often known as *attributes* or *properties*), and code, in the form of procedures (often known as *methods*). A feature of objects is an object's procedures that can access and often modify the data fields of the object with which they are associated (objects have a notion of "this" or "self"). In OOP, computer programs are designed by making them out of objects that interact with one another.[1][2] OOP languages are diverse, but the most popular ones are class-based, meaning that objects are instances of classes, which also determine their types.

Many of the most widely used programming languages (such as C++, Java, Python, etc.) are multi-paradigm and they support object-oriented programming to a greater or lesser degree, typically in combination with imperative, procedural programming. Significant object-oriented languages include Java, C++, C#, Python, PHP, JavaScript, Ruby, Perl, Object Pascal, Objective-C, Dart, Swift, Scala, Common Lisp, MATLAB, and Smalltalk.

Functional programming (often abbreviated FP) is the process of building software by composing **pure functions**, avoiding **shared state, mutable data,** and **side-effects**. Functional programming is **declarative** rather than **imperative**, and application state flows through pure functions. Contrast with object oriented programming, where application state is usually shared and colocated with methods in objects.

Functional programming is a **programming paradigm**, meaning that it is a way of thinking about software construction based on some fundamental, defining principles (listed above). Other examples of programming paradigms include object oriented programming and procedural programming.

b) NOPE!

c)
A programming **paradigm** is a style, or “way,” of programming. Some **languages** make it easy to **write** in some **paradigms** but not others. Never use the phrase “programming **language paradigm**.” A **paradigm** is a way of doing something (like programming), not a concrete thing (like a **language**).

What Are the Pros of OOP?

1. It allows for parallel development.
If you're working with programming teams, then each can work independently of one another once the modular classes have been worked out. That allows for a relative level of parallel development that wouldn't be available otherwise.
2. The modular classes are often reusable.
Once the modular classes have been created, they can often be used again in other applications or projects. At times, little-to-no modification is necessary for the next project as well. That gives a team more flexibility once they get beyond the initial start-up phase.
3. The coding is easier to maintain.
With OOP, because your coding base has been centralized, it is easier to create a maintainable procedure code. That makes it easier to keep your data accessible when it becomes necessary to perform an upgrade. This process also improves the security of the programming since high levels of validation are often required.

What Are the Cons of OOP?

1. It can be inefficient.
Object-oriented programming tends to use more CPU than alternative options. That can make it be an inefficient choice when there are technical limitations involved due to the size that it can end up being. Because of the duplication involved, the first-time coding can be more extensive than other options as well.
2. It can be too scalable.
If OOP is left to run out of control, then it can create a massive amount of bloated, unnecessary code. When that occurs, the overhead rises and that makes it difficult to keep costs down.
3. It can cause duplication.
OOP projects tend to be easier to design than implement. That is because of the modular classes are so flexible in their application. You may be able to get new projects up and running at a greater speed, but that comes at the cost of having projects sometimes feel like they've been cloned.

object-oriented languages example :

object-oriented languages include Java, C++, C#, Python, PHP, JavaScript, Ruby, Perl, Object Pascal, Objective-C, Dart, Swift, Scala, Common Lisp, MATLAB, and Smalltalk.

Is it possible to write object-oriented programs in C?

Yes, you **can**. People were writing **object-oriented C** before **C++** or Objective-**C** came on the scene. Both **C++** and Objective-**C** were, in parts, attempts to take some of the OO concepts used in **C** and formalize them as part of the language.

Q3:

```java
package com.company;

import java.util.Scanner;

public class Main {

    public static void main(String[] args) {
        Scanner sc = new Scanner(System.in);

        System.out.print("olgo string is : ");
        String olgo = sc.nextLine();

        System.out.print("str string is : ");
        String str = sc.nextLine();

        int i = 0, j = 0;
        int strLen = olgo.length();
        int lenght = olgo.length();
        int temp = 1;
        int stringLen = str.length();

        for (char s : olgo.toCharArray()) {

            if( s == '!' )
            {
                if( str.charAt(j - 1) == str.charAt(j) )
                {
                    System.out.println("true");
                    return;
                }
            }

            String check = null;
            if (s == '*') {
                check = olgo.substring(0, i);
                olgo = olgo.substring(i + 1, strLen); //recursive

                if (!str.contains(check)) { break; }

                int index = str.indexOf(check);
                str = str.substring(index + check.length(), stringLen);
                i = -1;
                strLen = olgo.length();
                stringLen = str.length();
            }

            if (temp == lenght) {
                check = olgo.substring(0, i + 1);
                olgo = olgo.substring(i + 1, strLen); //recursive

                if (!str.contains(check)) { break; }

                if( str.substring(stringLen - strLen ,stringLen).equals(check))
                {
```

```
                    System.out.println("true");
                    return;
                }
                int index = str.indexOf(check);
                str = str.substring(index + check.length(), stringLen);
                i = -1;
                strLen = olgo.length();
                stringLen = str.length();
            }

            if (strLen == 0 && stringLen == 0)
            {
                System.out.println("true");
                return;
            }
            if (olgo.length() == 0 && check.length() == 0 && stringLen != 0)
            {
                System.out.println("true");
                return;
            }
            i++;
            j++;
            temp++;

        }
        System.out.println("false!");
    }
}
```

this program removes substring before '*' and works like a recursive method.

Q4:

The **JRE** is the **Java Runtime Environment**. It is a package of everything necessary to run a compiled Java program, including the Java Virtual Machine (JVM), the Java Class Library, the `java` command, and other infrastructure. However, it cannot be used to create new programs.

The **JDK** is the **Java Development Kit**, the full-featured SDK for Java. It has everything the JRE has, but also the compiler (`javac`) and tools (like `javadoc` and `jdb`). It is capable of creating and compiling programs.

A **Java virtual machine** (**JVM**) is a virtual machine that enables a computer to run Java programs as well as programs written in other languages that are also compiled to Java bytecode. The JVM is detailed by a specification that formally describes what is required in a JVM implementation. Having a specification ensures interoperability of Java programs across different implementations so that program authors using the Java Development Kit (JDK) need not worry about idiosyncrasies of the underlying hardware platform.

Platform independent language means once compiled you can execute the program on any **platform** (**OS**). **Java** is **platform independent**. Because the **Java** compiler converts the source code to bytecode, which is Intermidiate Language. Bytecode can be executed on any **platform** (**OS**) using JVM(**Java** Virtual Machine).

default value of **primitive :**
In Java, the **default value** of any **primitive** is the 'zero' equivalent of that **primitive**. int,float,double are 0, char is \u0000, boolean is false. Since **primitives** are not actually objects on their face in Java, then they must be initialized by **default** because, semantically, they can never be null.

Q5:

```
package com.company;

import java.util.Scanner;

class SecondClass
{
    private int MyInt;

    public SecondClass(int i) {
        MyInt = i;
    }

    public void getMyInt()
    {
        Scanner sc = new Scanner(System.in);
        MyInt = sc.nextInt();
    }
}

public class Main
{
    public static void main(String[]args)
    {
        System.out.println("My First class");
        SecondClass sec = new SecondClass(2);
    }
}
```

Q6:
1) .class
2) object
3) Java.lang.Object class
4) constructor

Resources:

-Hacker.io
-wikipedia
-medium.com
-cs.lmu.edu
-stackoverflow
-javapoint