java assignment 1

1. Explain applications of OOP

Object-oriented programming (OOP) is a programming paradigm that focuses on creating objects that encapsulate data and behavior, and interact with one another to solve problems. Here are some common applications of OOP:

1. Graphical User Interfaces (GUIs): Many GUI applications are built using OOP concepts, with each GUI element represented as an object with properties and methods.
2. Games: OOP is commonly used in game development to model game elements as objects. For example, a game might have objects for players, enemies, weapons, and other elements.
3. Database Systems: OOP can be used to model the entities in a database system, with each entity represented as an object.
4. Web Applications: OOP is commonly used in web development frameworks such as Ruby on Rails and Django to model web entities as objects. For example, a web application might have objects for users, posts, comments, and other entities.
5. Simulation and Modeling: OOP can be used to model real-world systems such as physical simulations, financial modeling, and scientific simulations.
6. Artificial Intelligence: OOP is commonly used in AI systems to model knowledge representation and reasoning, and to represent agents that interact with their environment.
7. Mobile Applications: OOP is commonly used in mobile app development to model app elements as objects. For example, a mobile app might have objects for screens, buttons, text fields, and other elements.

These are just a few examples of how OOP is used in various applications. OOP's ability to encapsulate data and behavior into objects makes it a powerful tool for building complex, scalable, and maintainable software systems.

2. Explain various features of java

Java Comes with various distinct features as compared to other object oriented programming langues.

Object oriented: Java is an OOPL that supports the construction of programs that consist of collections of collaborating objects. These objects have a unique identity and have OOP features such as encapsulation, abstraction, Class, inheritance and polymorphism.

Simple: Java was designed with a small number of language constructs so that programmers could learn it quickly which make it simple. It eliminates several language features that were in C/C++ that was associated with poor programming practices or rarely used such as multiple inheritance, goto statements,

header files, structures, operator overloading, and pointers, security reason was also there for removing or not adding pointer in java.

Secure: Java is designed to be secure in a networked environment. The Java run-time environment uses a byte code verification process to ensure that code loaded over the network does not violate Java security constraints. Absence of pointer also adds on values for java security features.

Platform Independent: As we have seen on Introduction page Byte code and JVM which plays the major role for making the java platform independent and which make it a distinct.

Robust: Java is designed to eliminate certain types of programming errors. Java is strongly typed, which allows extensive compile-time error checking. Its automatic memory management (garbage collection) eliminates memory leaks and other problems associated with dynamic memory allocation and deallocation. It does not support memory pointers, which eliminates the possibility of overwriting memory and corrupting data.

Portable: has usually meant some work when moving an application program to another machine. Recently, the Java programming language and runtime environment has made it possible to have programs that run on any operating system and machine that supports the Java standard (from Sun Microsystems) without any porting work.

Architecture Neutral: Java applications that are compiled to byte codes can be interpreted by any system that implements the Java Virtual Machine. Since the Java Virtual 10. High Performan ce 11. Distribute d 9. Interpreted 7. Dynamic 5. Robust 3. Secured 1. Object Oriented 2. Simple 4. Platform Independe -nt 6. Portable 8. Architectu re Neutral 12.Multi Threded Features of Java Machine is supported across most operating systems, this means that Java applications are able to run on most platforms.

Dynamic: Java supports dynamic loading of classes class loader is responsible for loading the class dynamically in to JVM.

Interpreted: Java is compiled to bytecodes, which are interpreted by a Java run-time environment.

High Performance: Although Java is an interpreted language, it was designed to support “just-in-time” compilers, which dynamically compile bytecodes to machine code

Multithreaded: Java supports multiple threads of execution including a set of synchronization primitives. This makes programming with threads much easier.

Distributed: Java is designed to support various levels of network connectivity. Java applications are network aware: TCP/IP support is built into Java class libraries. They can open and access remote objects on the Internet.

3. What is Java Environment? Explain in brief?

Java combines both the approaches of compilation and interpretation. After compiling source file that is .java file compiler produces a .Class file which is nothing but a collection of byte code, at the run time, Java Virtual Machine (JVM) interprets this byte code and generates machine code which will be directly executed by the machine in which java program runs. So java is both compiled and interpreted language.

Byte code: As discussed above, javac compiler of JDK compiles the java source code into byte code so that it can be executed by JVM. The byte code is saved in a .class file by compiler.

Class Loader: The Java Class loader is a part of the Java Run-time Environment that is responsible for dynamically loading of Java classes into the Java Virtual Machine. The Java run time system does not need to know about files and file systems because of class loaders

Byte Code Verifier: Is again a part of Java Run-time Environment, after loading the byte code in JVM byte code are first inspected by a verifier. The verifier checks that the instructions cannot perform actions that are obviously damaging.

Java Virtual Machine (JVM): This is generally referred as JVM. Before, we discuss about JVM lets see the phases of program execution. Phases are as follows: we write the program, then we compile the program and at last we run the program.

1. Writing of the program is of course done by java programmer like you and me.
2. Compilation of program is done by javac compiler, javac is the primary java compiler included in java development kit (JDK). It takes java program as input and generates java byte code as output.
3. In third phase, JVM executes the byte code generated by compiler. This is called program run phase

4. Write difference between Java and C language?

C++	JAVA
C++ is platform- dependent	Java is platform-independent
It is used for system programming	It is used for programming in web-based, mobile or window applications.
It was the extension of C programming language.	It was designed for network computing.
It supports goto statement	Java does not.
Supported	Java doesn't support. It can be achieved using interface.
Supported	Not Supported
Supported	Supports pointers internally.
C++ uses compiler only.	Java uses compiler & interpreter both.

5. Explain Object and Class with real life example

1.7.1 Java Class: Class is the Entity which binds the all data members and data functions together.It can also be defined as class is the basic building block of an object-oriented language which is a template that describes the data and behaviour associated with instances of that class. When you instantiate a class you create an object that looks and feels like other instances of the same class.
Syntax to Write a Class:

```
Keyword class ClassName
{
\\ Variable Declarations

main function ()
{
\\Your Logic comes here
}
}
```

Saving Java File: Save a java File as ClassName.java
Is it Compulsory to save Java File as same name with Class Name? Answer is NO, When we save file and when we compile it then the compiler generates .class file with the same name as class name, and remember always for compilation we use .java file and for execution we use .class file. So if you want to save your program with different name than your class name, then you just have to run your .class file which is generated after compilation.
Example 1:

```
classHelloWorld
{
int i;   // Variable Declaration
public static void main(String [] args)   //Main Function Declaration
{
int j;
System.out.println(" Hello World Welcome to Java");
}
}
```

Output: Hello World Welcome to Java

6. What are the Data types in Java?

Primitive data types are those data types which are provided by the language in build such as int, char, float etc.A primitive data type specifies the size and type of variable values, and it has no additional methods.There are eight primitive data types in Java:

Data Type	Size	Description
byte	1 byte	Stores whole numbers from -128 to 127
short	2 bytes	Stores whole numbers from -32,768 to 32,767
int	4 bytes	Stores whole numbers from -2,147,483,648 to 2,147,483,647
long	8 bytes	Stores whole numbers from -9,223,372,036,854,775,808 to 9,223,372,036,854,775,807
float	4 bytes	Stores fractional numbers. Sufficient for storing 6 to 7 decimal digits
double	8 bytes	Stores fractional numbers. Sufficient for storing 15 decimal digits
boolean	1 bit	Stores true or false values
char	2 bytes	Stores a single character/letter or ASCII values

7. What is "System.out.println"?

"System.out.println" is a statement in Java that is used to print output to the console or terminal."System" is a predefined class in Java, which represents the system or environment in which the Java program is running. "out" is a static member of the System class that represents the standard output stream, which is typically the console or terminal.**"println" is a method of the "out" object that is used to print a line of text to the standard output stream, followed by a newline character. The text to be printed can be enclosed in double quotes or can be the value of a variable.****Here's an example of how "System.out.println" can be used to print a message to the console:**

System.out.println("Hello, world!");

When this statement is executed, the text "Hello, world!" will be printed to the console followed by a newline character.

8. What is Java Virtual Machine? Explain in brief?

**Java Virtual Machine (JVM): The Java Virtual Machine (JVM) is the virtual machine that runs the Java bytecodes. The JVM doesn't understand Java source code; that's why you need compile your *.java files to obtain .class files that contain the byte codes understood by the JVM. It's also the entity that allows Java to be a "portable language" (write once, run anywhere). Indeed, there are specific implementations of the JVM for different systems. The aim is that with the same byte codes they all give the same results.is a very important part of both JDK and JRE because it is contained or inbuilt in both. Whatever Java program you run using JRE or JDK goes into JVM and JVM is responsible for executing the java program line by line hence it is also known as interpreter.*

9. Explain Arithmetic operators in Java?

1.13.1 Arithmetic Operators

Arithmetic operators are used to perform common mathematical operations.

Operator	Name	Description	Example
+	Addition	Adds together two values	x + y
-	Subtraction	Subtracts one value from another	x - y
*	Multiplication	Multiplies two values	x * y
/	Division	Divides one value from another	x / y
%	Modulus	Returns the division remainder	x % y
++	Increment	Increases the value of a variable by 1	++x
--	Decrement	Decreases the value of a variable by 1	--x

10. What is Constructors? List and explain the types of constructors.

In Java, a constructor is a block of codes similar to the method. It is called when an instance of the object is created, and memory is allocated for the object. It is named as constructor because it constructs the values at the time of object creation. It is not necessary to write a constructor for a class. If your class doesn‟t have any constructor java compiler creates a default constructor for your class. you can say It is a special type of method which is used to initialize the object.

**

There are two type of constructor in Java:

a. No-argument constructor: A constructor that has no parameter is known as default constructor. If we don't define a constructor in a class, then compiler creates **default constructor(with no arguments)** for the class. And if we write a constructor with arguments or no-arguments then the compiler does not create a default constructor. Default constructor provides the default values to the object like 0, null, etc. depending on the type.

A constructor is called "Default Constructor" when it doesn't have any parameter.

Syntax of default constructor:<class-name>(){}

Example of default Constructor

//Java Program to create and call a default constructor//Java Program to create and call a default constructor

```
class Employee{

Employee()          //creating a default constructor
{
        System.out.println("Employee Class Default Constructor");

}
public static void main(String args[])    //main method
{
Employee b=new  Employee ();     //calling a default constructor
}

}
```

purpose of a default constructor

The default constructor is used to provide the default values to the object like 0, null, etc., depending on the type.

**

11. Explain Final class in brief.

In Java, a final class is a class that cannot be subclassed or extended. Once a class is marked as final, it cannot be inherited by any other class.

There are a few reasons why you might want to make a class final. One reason is to prevent the class from being modified or extended by other developers. This can be useful in situations where you want to ensure that the behavior of a class is consistent and predictable.

Another reason to make a class final is to improve performance. When a class is marked as final, the Java Virtual Machine (JVM) can optimize the code by inlining methods and eliminating unnecessary virtual method calls.

Here's an example of how to declare a final class in Java:
final class MyFinalClass { // class implementation }

In this example, the "MyFinalClass" class is marked as final, which means that it cannot be subclassed or extended by any other class.

It's worth noting that not all classes need to be marked as final. In fact, most classes in Java are not marked as final, as they are designed to be extended and customized by other developers. However, if you have a class that should not be modified or extended, marking it as final can be a useful tool.

12. State and explain the terms JDK, JRE & JVM.

1.5.1 Java Development Kit(JDK): As the name suggests this is complete java development kit that includes JRE (Java Runtime Environment), compilers and various tools like JavaDoc, Java debugger etc. In order to create, compile and run Java program you would need JDK installed on your computer.The Java Development Kit (JDK) is a software development environment used for developing Java applications and applets. It includes the Java Runtime Environment (JRE), an interpreter/loader (Java), a compiler (javac), an archiver (jar), a documentation generator (Javadoc) and other tools needed in Java development.

1.5.2 Java Runtime Environment(JRE):
The Java Runtime Environment (JRE) provides the libraries, the Java Virtual Machine, and other components to run applets and applications written in the Java programming language. In addition, two key deployment technologies are part of the JRE: Java Plug-in, which enables applets to run in popular browsers; and Java Web Start, which deploys standalone applications over a network. It is also the foundation for the technologies in the Java 2 Platform, Enterprise Edition (J2EE) for enterprise software development and deployment. The JRE does not contain tools and utilities such as compilers or debuggers for developing applets and applications. JRE includes JVM, browser plugins and applets support. When you only need to run a java program on your computer, you would only need JRE.

What does JRE consists of?
JRE consists of the following components:

- **Deployment technologies**, including deployment, Java Web Start and Java Plug-in.
- **User interface toolkits**, including Abstract Window Toolkit (AWT), Swing, Java 2D, Accessibility, Image I/O, Print Service, Sound, drag and drop (DnD) and input methods.
- **Integration libraries**, including Interface Definition Language (IDL), Java Database Connectivity (JDBC), Java Naming and Directory Interface (JNDI), Remote Method Invocation (RMI), Remote Method Invocation Over Internet Inter-Orb Protocol (RMI-IIOP) and scripting.
- **Other base libraries**, including international support, input/output (I/O), extension mechanism, Beans, Java Management Extensions (JMX), Java Native Interface (JNI), Math, Networking, Override Mechanism, Security, Serialization and Java for XML Processing (XML JAXP).
- **Lang and util base libraries**, including lang and util, management, versioning, zip, instrument, reflection, Collections, Concurrency Utilities, Java Archive (JAR), Logging, Preferences API, Ref Objects and Regular Expressions.
- **Java Virtual Machine (JVM)**, including Java Hotspot Client and Server Virtual Machines.

1.5.3Java Virtual Machine (JVM): The **Java Virtual Machine** (JVM) is the virtual machine that runs the Java bytecodes. The JVM doesn't understand Java source code; that's why you need compile your *.java files to obtain *.class files that contain the byte codes understood by the JVM. It's also the entity that allows Java to be a "portable language" (*write once, run anywhere*). Indeed, there are specific implementations of the JVM for different systems. The aim is that with the same byte codes they all give the same results.is a very important part of both JDK and JRE because it is contained or inbuilt in both. Whatever Java program you run using JRE or JDK goes into JVM and JVM is responsible for **executing the java program line by line** hence it is also known as interpreter.

13. What is garbage collection in java explain in brief.

Java garbage collection is the process by which Java programs perform automatic memory management. Java programs compile to bytecode that can be run on a Java Virtual Machine or JVM for short. When Java programs run on the JVM, objects are created on the heap, which is a portion of memory dedicated to the program. Eventually, some objects will no longer be needed. The garbage collector finds these unused objects and deletes them to free up memory.

Garbage Collection is process of reclaiming the runtime unused memory automatically. In other words, it is a way to destroy the unused objects.

1.14. qAdvantage of Garbage Collection

- It makes java **memory efficient** because garbage collector removes the unreferenced objects from heap memory.
- It is **automatically done** by the garbage collector(a part of JVM) so we don't need to make extra efforts.

OR

Garbage collection in Java is the automatic process of freeing up memory that is no longer being used by the Java program. Java's garbage collector runs periodically in the background, identifying objects in memory that are no longer being referenced by the program, and freeing up the memory allocated to those objects.

The garbage collector in Java works by tracking the references to objects in memory. When an object is created, it is assigned a reference that points to its location in memory. As long as there are references to that object, it will remain in memory. However, when there are no more references to an object, the garbage collector identifies it as garbage and frees up the memory allocated to it.

The garbage collector in Java is highly configurable, and there are various options available for controlling how it operates. For example, you can set the amount of memory that is allocated to the Java program, the frequency at which the garbage collector runs, and the criteria used for identifying objects that are no longer being used.

One of the advantages of garbage collection in Java is that it helps to prevent memory leaks. Memory leaks occur when an application allocates memory for an object but fails to release it when the object is no longer needed. This can cause the program to consume increasingly more memory over time, which can eventually lead to crashes or other performance issues. Garbage collection helps to prevent memory leaks by automatically releasing memory that is no longer being used, reducing the risk of running out of memory or crashing the program.