Amith. N
231039022

1) Discuss the object oriented programming concepts?

=> * It is a programming paradigm which revolves around the concept of objects.

* Objects are instances of classes & they encapsulate both data (attribute and behaviour).
* Java is popular language that embraces OOP principle.

① Classes and objects:

* Class is a blueprint for creating objects.
* Object is an instance of class.

Ex:
```
// Example of class
public class car {
    string name;
    string model;
public void start() {
    System.out.println("starting car");
}
}

//Creating objects of class
Car myCar = new Car();
myCar.name = "toyota";
myCar.model = "Corysta";
```

② Inheritance:

* Important pillar of OOP.
* Mechanism in which one class is allowed to inherit the features of another class.
* Creating new classes based on existing ones.
* It is used for code - reusability.
* Extend keyword is used to derive from existing class.

Syntax:

sub class — super class

```
class derived_class extends base_class
{
   // methods and fields.
}
```

```
// Example of inheritance
Class electricCar extends Car {
   int batteryCapacity;
   public void charge() {
      System.out.println("Charging the car");
      }
   }
ElectricCar myelectricCar = newelectricCar();
myelectricCar.name = "Tesla";
myelectricCar.model = "model 3";
myelectricCar.change();
```

(iii) Polymorphism:

* It allows objects of diffrent classes to be treated as objects of common superclass.
* Polymorphism means having many forms.
* It allows us to perform single action in diffrent ways.
* It allows you to define one interface & have multiple implementations.

```
// Example of polymorphism.
Car someCar = newelectricCar();
someCar.start(); // Calls electric car in start method.
```

(iv) Encapsulation:

* It is the practice of hiding internal details of an object & providing access through methods.
* It is achieved by declaring the instance variable of a class as private. They can only be accessed within the class.

```
//Example.
public class Student {
    private string name;
    private int age;

    public string getname() {
        return name;
    }
    public void setname(string name) {
        this.name = name;
    }
    public int getage() {
        return age;
    }
    public void setage(int=age)
        this.age = age
    }
    }

public class main {
    public static void main(string[] args) {
        student.student = new Student();
        student.setname = ("John");
        student.setage(20);

        System.out.println(student.getname());
        System.out.println(student.getAge());
        }
    }
```

⑤ Abstraction:

* It is the process of hiding the implementation details of an object & only exposing essential features to user.
* It can be achieved using abstract classes & interfaces.
* Abstract class is a class that won't be instantiated
* Abstract classes can contain abstract methods, which are method a, that do not have body.

```
// Example:
abstract class shape {
    abstract double area();
}
class circle extends shape {
    double radius;

@Override
double area() {
    return Math.PI * radius * radius;
    }
}
Circle circle = new circle();
circle.radius = 5.0;
double circle area = circle.area();
```

2) Distinguish between OOP's and procedure oriented programming.

Feature	OOP's programming	Procedure oriented programming
Approach	Organised around objects & classes.	Organised as a squence of functions / steps.
Data abstraction	Encourages encapsulation & data hiding.	Data & functions may be sperate.
Inheritance	Supports inheritance for code reuse.	Not supported
Polymorphism	Encourages polymorphism for flexibility.	Not supported

3) Explain difference between Java applets and applications.

Feature	Java applet	Java application
Execution	Inside a web browser	On the user's computer
Security	Less secure	More secure
Deployment	As part of web page	As a standalone program
Capabilities	Limited	Not limited
Advantages	Small, platform independent, easy to develop & display.	Powerfull, versatile
Disadvantages	Less secure, limited capabilities not as widely used.	Not as small & platform independent not as easy to develop and display.

4) Discuss the features of Java.

* Object oriented programming: This is based on the concept of objects, this makes java code easy to read & understand.

* Java programs are platform independent, which means that they can run on any computer.

* Java has built-in security features help to protect against common security threats such as worms.

* Java is a robust language help to prevent errors and crashes, such as garbage collection & exception handling.

* Java is a multithreaded language, it can run multiple tasks at same time.

* Java is high performance language used to develop applications that are both efficient & scalable.

* Java is dynamic language used to create applications that can change at runtime.

* Java provides reflection, which allow you to inspect & modify behaviour of classes & objects at runtime.

* Java provides generics, which allow you to write code that can work with different types of data.

5) Write a short note on following.

(a) JVM:

* It plays a pivotal role in making java a platform indepen-
-end language.
* Java source code is compiled into an intermediate binary format known as bytecode.
* JVM responsible for interrupts / compiling bytecode into native machine code executed by host system.
* JVM manages the memory allocation & garbage collection helps prevent memory leaks & enchances application stability.
* JVM includes built-in security features to safeguard against unauthorized access & malicious code.
* JVM comes with a comprehensive set of standard libraries known as java standard library.

(b) Byte code:

* Its an intermediary representation of java source code generated during compilation process.
* Bytecode is a compact binary format making, it efficient for storage & transmission.
* Bytecode can only be executed within the controlled environ-
-ment of JVM providing level of security.
* Bytecode can be decompiled into human-readable form which can be usefull for debugging.
* Bytecode serves as an intermediate step between the java source code & native machine code.

6) How java achieves platform independent codes?

⇒ java achieves platform independence through the use of bytecode & java virtual machine (JVM).

(a) Bytecode:

* when you compile a java source code file, the java compiler translates into binary format called bytecode.
* bytecode saved in class files.

(b) JVM:

* Software based runtime environment executes java bytecode.
* When you run a java program, jvm interprets @ compile bytecode into native machine code.

(c) Platform specific JVM's:

* To achieve platform independence - java developer creates platform specific implementation of JVM.

(d) Standard library (Java API):

* java provides a comprehensive standard library known as java API.
* This library includes vast collection of classes & methods.

(e) No native code:

* java doesn't allow direct execution of native machine code, a system specific calls.

(f) Security & sandboxing:

* java's security features such as class loader & bytecode verification, ensure that untrusted code cannot harm host system.

(g) Write once, run anywhere (WORA):

* Combination of bytecode, the JVM, & java API enables "WORA".

7) What do you mean by just in time compilation? (5)

* JIT compilation is a technique used in many programming languages, including java to improve performance of programs by dynamically translating intermediate code.
* JIT compilation combines benefits of both interpretates and compilation.
* JIT compilation can lead to significant performance improvements.
* JIT compilers are selective, they analyze bytecode & identify frequency executed code.
* JIT compilers are used with adaptive optimization technique which means they continue to monitor program execution & can recompile code.
* JIT compilation helps maintain java's platform independence