K.Varsha
B191437
Roll No:27
E3-CSE-C1 (311)

Object Oriented Programming

Assignment-2

1. Define classes and objects in java. Explain the concept of inheritance, including single, multilevel & hierarchial inheritance along with method Overriding?

classes and objects

In java, class is a blue print or a template for creating objects. Objects are instance of classes. A class defines the properties and behaviour (methods) that its objects will have.

Example:

```
 //Defining a class
 class car {
     // fields
     String make;
     String model;
     int year;
     //method
     void start(){
         System.out.Println("The car is starting:");
     }
 }
 //creating an object of the class car
 car myCar=new car();
  myCar.make = "Toyota";
  myCar.model = "Comry";
  myCar.year = 2022;
  myCar.start();
```

Inheritance:

Inheritance is a fundamental concept in Object Oriented programming that allows a new class (Sub class (or) derived class) to inherit properties and the behaviour of an existing class

There are several types of inheritance:

1) Single Inheritance: class Animal {

```
        void at(){
          S.O.P("Dog is bar King");
        }
    }
```

2) Multilevel inheritance

```
 class A{
      Void method(){
           S.O.P("Method of class A");
      }
 }
 class B extends A{
      void method(){
          System.out.Println(" Method of class B");
      }
 }
 class C extends B{
      void method(){
           System.out.Println("Method of class C");
      }
 }
 Animal{
      Void Sound(){
          System.out.Println(" Animal make a Sound");
      }
 }
 class Dog extends Animal{
      Void Sound(){
          System.out.println("Dog Barks");
      }
 }
```

In this example dog class overrides the Sound() method, which is defined in Super class Animal, when you call Sound() on dog object, it will execute an Overridden method.

Overriding allows polymorphism.

2) Discuss the concept of polymorphism, including static & dynamic polymorphism. method overloading & the importance of visibility control (public, protected, default) in java?

polymorphism is fundamental concept in OOP that allows object of different type of treated as object of a common type. This simplifies code & prompts flexibility & enables single interrupt to represent various type.

Method Overloading: It is a form of compile time, static polymorphism where multiple method in the same class share the same name but have different parameter lists the compiler determines which method to call based on number & type of arguments.

```
    public class calculator{
        public int add (int a, int b){
            return a+b;
        }
    }
```

Method Overriding: It is a form of unique (dynamic) polymorphism when a subclass provides a specific implementation of a method that is already defined in its super class.

Ex:
```
public class Animal{
    public void makeSound(){
        System.out.println("Animal makes a Sound");
    }
}
public ~~Dog~~ class Dog extends Animal{
    public void makeSound(){
        S.O.P("Dog barks");
    }
}
```

Visibility control: Visibility control, archieved through access modifiers (public, Private, Protected, defaults) is crucial for encapsulation and controlling the access to class members.

Ex:
```
class A{
    Void method A(){
        System.out.println("Method of class A");
    }
}
class B extends A{
    Void method B(){
        System.out.println("Method of class B");
    }
}
class C extends B{
    void method(){
        System.out.println("Method of class C");
    }
}
```

3) Hierarchial Inheritance

Multiple class inherit from a Single Super class.

Ex.
```
class Shape{
        Void draw(){
            System.out.println("Drawing Shape");
        }
}
class circle extends Shape{
        Void drawCircle(){
            System.out.println("Drawing a circle");
        }
}
class Circle extends Shape{
        Void drawSquare(){
            System.out.println("Drawing a Square");
        }
}
```

4) Elaborate on the Significance of packages in java, naming conventions, creating accessing and hiding classes with in packages. Discuss the basics of the exception handling, its advantages and try catch blocks?

Packages in Java:

Significance:

1) Organizing code: package provide a way to organize and structure code by grouping related classes and interfaces together.

2) Namespace management: packages help prevent naming conflicts by Providing a namespace for the classes and interfaces together.

3) Access Control:
Packages allow you to control access to classes and members using access modifiers.

Naming conventions:

1. LowerCase: package names are typically in lower case letter.
2. Uniquenames package name should be unique to avoid naming conflicts.
3. Reverse Domain: It's a Domain common convention to use reverse domain name as the package name.

Underscores & Hyphens:
Package name should not contain underscores or hypothesis.

Creating packages:

```
package com.example.myProjects;
public class myClass(){
    // class implementation
}
```

Accessing classes within Packages:

```
import com.example.myProject.myClass;
Public class AnotherClass {
   myclass myObject = new myClass;
}
```

Hiding classes within packages:

```
package com.example.myProject;
class HiddenClass {
        // class implementation
}
```

Exception Handling:

Basics: Exception handling is a mechanism in java that deals with runtime errors.

try block: contains the code that might throwon an exception.

catch block: catches & handles the exceptions thrown in the try Block.

finally block: contains a code that is always executed whether an exceptions occurs or not.

Try-catch blocks:

```
try {
   //code that might throw an Exception
} catch (exceptionType1 e) {
   //handle exception of type exceptionType1
}
catch (exceptionType2 e) {
   //handle exception of type exceptionType2
}
finally {
   //code that always executes regardless of exceptions occurence
}
```

5) Explain the concepts of multi-threading, thread life cycle, Synchronization & thread Scheduling in Java. Discuss the different types of I/O Streams and their advantages?

Multithreading in Java:

Thread: A thread is the smallest unit of execution in a program. In java, you can create and manage thread using the 'thread' class or the 'Runnable' interface.

Thread life cycle:

New: The thread when it is created.

Runnable: The thread is read to run but the Schedule has not started.

Blocked: The thread is waiting for a monitor lock & cannot proceed.

waiting: The thread is waiting for another thread to perform a particular tasks for a Specified period.

Time waiting:

The thread is waiting for another thread for a Specified time period to Perform a particular action.

Terminated:

The thread has completed execution.

Synchronization:

→Synchronization in java is the process of controlling the access to shared resources by multiple threads to avoid data inconsistency & conflicts.

→It can be achieved using the 'Synchronized' Keyword (or) using explicit locks from the java.util. concurrent. package.

Thread Scheduling:

→The java virtual machine (JVM) uses a thread schedule to determine which thread should run.

→ Thread priorities (defined by "Thread.MAX_PRIORITY") ; Influence Scheduling.

I/O stream in Java:

Streams: In java, input & output are based on the concept of streams.

A stream is a Sequence of data elements accessed sequentially.

Types of streams:

Byte streams: "Input Stream" and "Output Stream" are the base classes for the byteStream input and output.

character stream:

Reader & writer are the base classes for character Stream input & Output.

Object Stream:

Object Input stream and Object Output stream allow the Serialization & Serialization of java objects.

Buffered Streams:

Buffered InputStream & Buffered Output stream provide buffering capabilities improving the efficiency of I/o operations.

Advantages of I/o streams:

1. Abstraction: Streams provide a high-level abstraction making it easy to work with different types of data Source & destinations.

2. Efficiency: Buffered streams enhance I/o performance by ~~reading~~ reducing the no.of actual I/o operations.

3. Flexibility:

Streams Support Various data types and can be easily chained /combined to perform complex I/o operations.

4) Serialisation:

Object streams facilitate to Serialization of java objects, allowing the persistence of the object states.

5. Uniformity:

Whether handling bytes (or) character, Stream provide a uniform and consistent approach to I/o operations.