NAME- Harshita Totala ROLL NO- SEITA14 DIV- IT A

SUBJECT- Object Oriented Programming

ASSIGNMENT NO: 2

Title – Polymorphism.

Aim - Identify commonalities and differences between Publication, Book and Magazine classes.Title, Price, Copies are common instance variables and saleCopy is common method. Thedifferences are, Book class has author and orderCopies(). Magazine Class has orderQty,Currentissue, reciveissue(). Write a program to find how many copies of the given books areordered and display total sale of publication.

Objective - To learn the concept of Polymorphism.

Theory- This section explains the Object Oriented Concept, Polymorphism. Polymorphism is the ability of an entity to behave in different forms. Take a real-world example; the Army Ants. There are different size of ants in the same ant colony with different responsibilities; workers are in different sizes and the queen is the largest one. This is a polymorphic behavior where the entities have a unique feature while they share all other common attributes and behaviors.

Polymorphism is considered as one of the important features of Object Oriented Programming. Polymorphism allows us to perform a single action in different ways. In other words, polymorphism allows you to define one interface and have multiple implementations. The word “poly” means many and “morphs” means forms, So it means many forms. There are two types of polymorphism in java:

1) Static Polymorphism also known as compile time polymorphism
2) Dynamic Polymorphism also known as runtime polymorphism

1]Compile time Polymorphism or Static polymorphism:-

→ Polymorphism that is resolved during compiler time is known as static polymorphism. Method overloading is an example of compile time polymorphism.

Method Overloading: This allows us to have more than one method having the same name, if the parameters of methods are different in number, sequence and data types of parameters. Example : Method overloading is one of the way java supports static polymorphism. Here we have two definitions of the same method add() which add method would be called is determined by the parameter list at the compile time. That is the reason this is also known as compile time polymorphism. class SimpleCalculator

```
{
int add(int a,int b)
 {
return a+b;
}
int add(int a,int b,int c)
{
return a+b+c;
}
}
public class Demo
{
public static void main(String args[])
{
SimpleCalculator obj=new SimpleCalculator();
System.out.println(obj.add(10,20));
System.out.println(obj.add(10,20,30));
}
}
```

2]Runtime Polymorphism or Dynamic polymorphism:-

⟶ It is also known as Dynamic Method Dispatch. Method overriding is an example of runtime polymorphism. Dynamic polymorphism is a process in which a call to an overridden method is resolved at runtime, thats why it is called runtime polymorphism. Method Overriding: Declaring a method in sub class which is already present in parent class is known as method overriding. Overriding is done so that a child class can give its own implementation to a method which is already provided by the parent class. In this case the method in parent class is called overridden method and the method in child class is called overriding method.

Method Overriding Example: We have two classes: A child class Boy and a parent class Human.The Boy class extends Human class. Both the classes have a common method void eat (). Boy class is giving its own implementation to the eat () method or in other words it is overriding the eat () method.

The purpose of Method Overriding is clear here. Child class wants to give its own implementation so that when it calls this method, it prints Boy is eating instead of Human is eating. class Human{ //Overridden method

```
public void eat()
{
System.out.println("Human is eating");
}
}
Class Boy extends Human{ //Overriding
method
public void eat(){

System.out.println("Boy is eating");
}
```

```
public static void main(String args []) {
Boy obj=new Boy();
//This will call the child class version of eat() obj.eat();
}
}
```
Output:
Boy is eating

Advantage of method overriding: The main advantage of method overriding is that the class can give its own specific implementation to a inherited method without even modifying the parent class code. This is helpful when a class has several child classes, so if a child class needs to use the parent class method, it can use it and the other classes that want to have different implementation can use overriding feature to make changes without touching the parent class code.
Method Overriding is an example of runtime polymorphism. When a parent class reference points to the child class object then the call to the overridden method is determined at runtime, because during method call which method (parent class or child class) is to be executed is determined by the type of object. This process in which call to the overridden method is

```
class ABC{ //Overridden
method
public void disp()
{
System.out.println("disp() method of parent class");
}
}
class Demo extends ABC{
//Overriding          method
public void disp(){
System.out.println("disp() method of Child class");
}
public void new Method(){
System.out.println("new method of child class");
}
public static void main(String args []){
/* When Parent class reference refers to the parent class object *
then in this case overridden method (the method of parent class) *
is called.
*/
ABC obj=new ABC();

obj.disp();
/* When parent class reference refers to the child class object *
then the overriding method (method of child class) is called.
* This is called dynamic method dispatch and runtime polymorphism
*/
ABC obj2 =new Demo(); obj2.disp();
```

```
}
}
```

Output: disp() method of
parent class
disp() method of Childclass

In the above example the call to the disp() method using second object (obj2) is runtime polymorphism In dynamic method dispatch the object can call the overriding methods of child class and all the non-overridden methods of base class but it cannot call the methods which are newly declared in the child class.

In the above example the object obj2 is calling the disp(). However if you try to call the newMethod() method (which has been newly declared in Demo class) using obj2 then you would give compilation error.

Rules of method overriding in Java:-

Rule 1: Overriding method name and the overridden method name must be exactly same.

Rule 2: Overriding method must have the same set of parameters as the overridden method.

Rule 3: The return type of overriding method name must be same as the super class's method.

Rule 4: Access modifier of the overriding method must be same or less restrictive than the overridden method's access modifier.

Rule 5: The overriding method can throw new unchecked exceptions but cannot throw new checked exceptions.

Program-

```
import java.util.Scanner; class
publication
{
    Scanner sc = new Scanner(System.in);
    String title;
    int p, cp, quantity;

    void input()
    {
        System.out.print("Enter title: ");
title = sc.nextLine();
        System.out.print("Enter price: ");
p = sc.nextInt();
        System.out.print("Enter copies: " );
        cp = sc.nextInt();
    }
}
class book extends publication
{
    Scanner sc = new Scanner(System.in);
    String author;

    void orderCopies() {
input();
        System.out.print("Enter Author of book: ");
author = sc.nextLine();
        System.out.print("Enter quantity of books you want-: ");
quantity = sc.nextInt();
    }
    void saleCopy()
    {
        System.out.println("\n------*** DETAILS OF BOOKS ***-------");
        System.out.println("BOOK NAME:- " + title);
        System.out.println("AUTHOR NAME:- " + author);
        System.out.println("PRICE:- "+ p+" Rs");
        System.out.println("TOTAL SALE OF BOOK:- " + ((quantity+cp) *p)+ " Rs");
        System.out.println("---------------------------------------");
}
}

class magazine extends publication
{
    Scanner sc = new Scanner(System.in);
    String c_issue;
```

```
    void saleCopy()
    {
        System.out.println("\n------*** DETAILS OF MAGAZINE ***-------");
        System.out.println("MAGAZINE NAME:- " + title);
        System.out.println("CURRENT ISSUE:- " + c_issue);
        System.out.println("PRICE:- " + p + " Rs");
        System.out.println("TOTAL SALE OF MAGAZINE:- " + ( (quantity+cp) * p) + " Rs");
        System.out.println("----------------------------------------");
    }
    void currentIssue()
    {
input();
        System.out.print("Enter current issue: " );
c_issue = sc.next();
    }
    void orderQty()
    {
        System.out.print("Enter quantity of magazine: ");
        quantity=sc.nextInt();
    }
}

public class polymorphism{     public
static void main(String[] args)
    {
        publication p= new publication();
System.out.println("\n----BOOK SELL----");
book b = new book();         b.orderCopies();
        b.saleCopy();

        System.out.println("\n----MAGAZINE SELL----");
magazine m = new magazine();         m.currentIssue();
        m.orderQty();
        m.saleCopy();
    }
    }
```

polymorphism.java

```java

//Assignment 2- Polymorphism.

import java.util.Scanner;
class publication
{
    Scanner sc = new Scanner(System.in);
    String title;
    int p, cp, quantity;

    void input()
    {
        System.out.print("Enter title: ");
        title = sc.nextLine();
        System.out.print("Enter price: ");
        p = sc.nextInt();
        System.out.print("Enter copies: " );
        cp = sc.nextInt();
    }
}
class book extends publication
{
    Scanner sc = new Scanner(System.in);
    String author;

    void orderCopies() {
        input();
        System.out.print("Enter Author of book: ");
        author = sc.nextLine();
        System.out.print("Enter quantity of books you want-: ");
        quantity = sc.nextInt();
    }
    void saleCopy()
    {
        System.out.println("\n------*** DETAILS OF BOOKS ***--------");
        System.out.println("BOOK NAME:- " + title);
        System.out.println("AUTHOR NAME:- " + author);
        System.out.println("PRICE:- "+ p+" Rs");
        System.out.println("TOTAL SALE OF BOOK:- " + ((quantity+cp) *p)+ " Rs");
        System.out.println("----------------------------------------");
    }
}

class magazine extends publication
{
    Scanner sc = new Scanner(System.in);
    String c_issue;

    void saleCopy()
    {
        System.out.println("\n------*** DETAILS OF MAGAZINE ***-------");
        System.out.println("MAGAZINE NAME:- " + title);
        System.out.println("CURRENT ISSUE:- " + c_issue);
        System.out.println("PRICE:- " + p + " Rs");
        System.out.println("TOTAL SALE OF MAGAZINE:- " + ( (quantity+cp) * p) + " Rs");
        System.out.println("----------------------------------------");
    }
    void currentIssue()
    {
        input();
```

```
        System.out.print("Enter current issue: " );
        c_issue = sc.next();
    }
    void orderQty()
    {
        System.out.print("Enter quantity of magazine: ");
        quantity=sc.nextInt();
    }
}

public class polymorphism{
    public static void main(String[] args)
    {
        publication p= new publication();
        System.out.println("\n----BOOK SELL----");
        book b = new book();
        b.orderCopies();
        b.saleCopy();

        System.out.println("\n----MAGAZINE SELL----");
        magazine m = new magazine();
        m.currentIssue();
        m.orderQty();
        m.saleCopy();
    }
    }
```

Output-

```
Run:  polymorphism

----BOOK SELL----
Enter title: My Journey
Enter price: 650
Enter copies: 10
Enter Author of book: A.P.J Abdul Kalam
Enter quantity of books you want-: 5

------*** DETAILS OF BOOKS ***--------
BOOK NAME:- My Journey
AUTHOR NAME:- A.P.J Abdul Kalam
PRICE:- 650 Rs
TOTAL SALE OF BOOK:- 9750 Rs
---------------------------------------

----MAGAZINE SELL----
Enter title: Times of India
Enter price: 150
Enter copies: 6
Enter current issue: May
Enter quantity of magazine: 4

------*** DETAILS OF MAGAZINE ***--------
MAGAZINE NAME:- Times of India
CURRENT ISSUE:- May
PRICE:- 150 Rs
TOTAL SALE OF MAGAZINE:- 1500 Rs
----------------------------------------

Process finished with exit code 0
```

Conclusion- Hence, we have studied the concept of method overriding, method overloading and Polymorphism.