

Object Oriented Programming in Java Laboratory

Name of Student: Rishabh Jain

Roll Number: SE21IT047

PRN No:72139725F

Batch: A2

Lab Coordinators : (Mrs. Sapna Kolambe)

Assignment No: -02

Department: -Information Technology

Batch: -A2

Roll Number: Second Year-IT047

Date of Assignment: -

Assignment: 2

Title: Identify commonalities and differences between Publication, Book and Magazine classes.

Title, Price, Copies are common instance variables and saleCopy is common method. The differences are, Book class has author and orderCopies(). Magazine Class has orderQty, Currentissue, receiveissue(). Write a program to find how many copies of the given books are ordered 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

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));
```

```
}
```

```
}
```

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, that's 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.

```
classHuman{
//Overridden method
publicvoid eat()
{
System.out.println("Human is eating");
}
}
classBoyextendsHuman{
//Overriding method
publicvoid eat(){

System.out.println("Boy is eating");
}
publicstaticvoid main(Stringargs[]){
Boyobj=newBoy();
//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
publicvoiddisp()
{
System.out.println("disp() method of parent class");
}
}
classDemoextends ABC{
//Overriding method
publicvoiddisp(){
System.out.println("disp() method of Child class");
}
publicvoidnewMethod(){
System.out.println("new method of child class");
}
publicstaticvoid main(Stringargs[]){
/* 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=newABC();
```

```
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 =newDemo();
```

```
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.

Method Overriding with example (Run time Polymorphism):

Advantages of method overriding:

Sample Code:

Consider Book & Magazines both specific type of publication

Attribute title, author & price are obvious parameter. For Book, orderCopies() takes parameter specifying how

many copies are added to stock. For Magazine, orderQty is number of copies received of each new issue and

currIssue is date/period of current issue. We can separate out these common member of classes into superclass

called Publication. The differences will need to be specified as additional member for the 'subclasses' Book

and Magazine.

```
public class Publication {
```

```
// Title of the publication.
```

```
private String title;
```

```
// Price of the publication.
```

```
private double price;
```

```
// copies of the publication.
```

```
private int copies;
```



```

public String getTitle() {
    return this.title;
}

public void setTitle(String title) {
    this.title = title;
}

public void setPrice(double price) {
    if (price > 0) {
        this.price = price;
    } else {
        System.out.println("Invalid price");
    }
}

public double getPrice() {
    return this.price;
}

public int getCopies() {
    return this.copies;
}

public void setCopies(int copies) {
    this.copies = copies;
}

public void sellCopy(int qty) {
    System.out.println("Total Publication sell: $" + (qty * price));
}

}

public class Book extends Publication {
    // Author of the book.
    private String author;

    public String getAuthor() {
        return this.author;
    }
}

```

```

}

public void setAuthor(String author) {
    this.author = author;
}

Public void ordercopies(intpcopies){
    Setcopies(getcopies() + pcopies);
}

public void sellcopy(intqty) {
    System.out.println("Total Book sell: $" + (qty * price));
}

}

public class Magazine extends Publication {
    privateintorderQty;
    private String currIssue;
    public String getcurrIssue() {

        returnthis.currIssue;
    }

    public void setcurrIssue(String issue) {
        this.currIssue = issue;
    }

    publicintgetorderQty() {
        returnthis.orderQty;
    }

    public void setorderQty(int copies) {
        this.orderQty = copies;
    }

    public void sellcopy(intqty) {
        System.out.println("Total Magazine sell: $" + (qty * price));
    }

    Public void recvNewIssue(string pNewIssue){

```

```

Setcopies(orderQty);

currIssue=pNewIssue;

}

}

Class mainClass{

Public static void main(String [] args)

{

//accept all details of book to be order such as title, author, price & copies;

Book obj1 = new Book();

Obj1.ordercopies(copies);

Publication obj2 = new Book();

Obj2.sellcopy(copies); //Overriden method is invoke

Publication obj3 = new Publication();

Obj3.sellcopy(copies);

}

```

Code:

```

package com.company;

import java.util.Scanner;

abstract class Publication {
    String title;
    int price ,copies ,quantity;
    abstract void saleCopy();

    void setprice(){
        Scanner sc = new Scanner(System.in);
        System.out.print("Enter the title :");
        title = sc.next();
        System.out.print("Enter the price :");
        price = sc.nextInt();
        System.out.print("Enter the copies :");
        copies = sc.nextInt();
    }
    void display(){
        System.out.println(title+"\t"+price+"\t"+copies);
    }
}

class Book extends Publication{
    String author;

```

```

    void saleCopy() {
        System.out.println("*****");
        System.out.println("Book name :"+title);
        System.out.println("Author name :"+author);
        System.out.println("Price per book :"+price);
        System.out.println("Copies ordered :"+copies);
        System.out.println("\n *** Total sale  :
"+(copies+quantity)*price);
    }
    void setAuthor() {
        Scanner sc = new Scanner(System.in);
        System.out.print("Author Name :");
        author = sc.next();
    }

    void orderQty() {
        System.out.println("Enter the quantity of the books :");
        Scanner sc=new Scanner(System.in);
        quantity=sc.nextInt();
    }
    void displayBook() {
        display();
        System.out.println("\t"+author);
    }
}

class Magazine extends Publication{
    String currIssue;

    void saleCopy() {
        System.out.println("*****");
        System.out.println("Magazine name :"+title);
        System.out.println("Price per book :"+price);
        System.out.println("Copies ordered :"+copies);
        System.out.println("\n *** Total sale  :
"+(copies+quantity)*price);
    }
    void issue() {
        Scanner sc = new Scanner(System.in);
        System.out.println("Enter the issue :");
        currIssue=sc.next();
    }

    void orderQty() {
        System.out.println("Enter the quantity of magazine :");
        Scanner sc = new Scanner(System.in);
        quantity=sc.nextInt();
    }

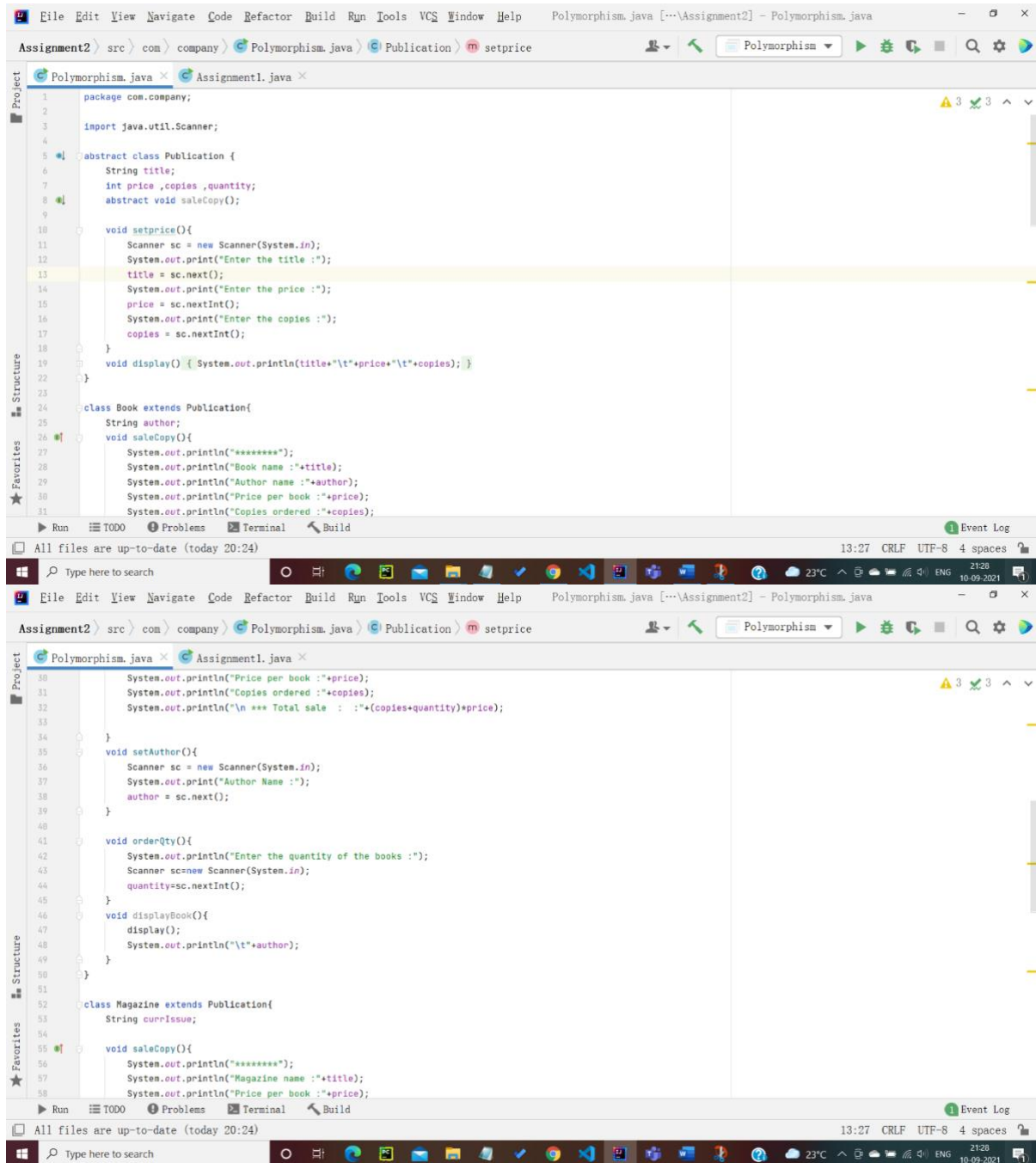
    void display_magazine() {
        display();
    }
    void recieveIssue() {
        System.out.println("You will recieve magazine in :"+currIssue);
    }
}

public class Polymorphism {

```

```
public static void main(String[] args) {  
    Book b=new Book();  
    b.setprice();  
    b.setAuthor();  
    b.orderQty();  
    b.saleCopy();  
  
    Magazine m = new Magazine();  
    m.setprice();  
    m.issue();  
    m.receiveIssue();  
    m.orderQty();  
    m.saleCopy();  
}  
}
```

OUTPUT :



```
package com.company;

import java.util.Scanner;

abstract class Publication {
    String title;
    int price ,copies ,quantity;
    abstract void saleCopy();

    void setprice(){
        Scanner sc = new Scanner(System.in);
        System.out.print("Enter the title :");
        title = sc.next();
        System.out.print("Enter the price :");
        price = sc.nextInt();
        System.out.print("Enter the copies :");
        copies = sc.nextInt();
    }

    void display() { System.out.println(title+"\t"+price+"\t"+copies); }
}

class Book extends Publication{
    String author;
    void saleCopy(){
        System.out.println("*****");
        System.out.println("Book name :"+title);
        System.out.println("Author name :"+author);
        System.out.println("Price per book :"+price);
        System.out.println("Copies ordered :"+copies);
    }
}

System.out.println("Price per book :"+price);
System.out.println("Copies ordered :"+copies);
System.out.println("\n *** Total sale : :"+(copies*quantity)*price);
}

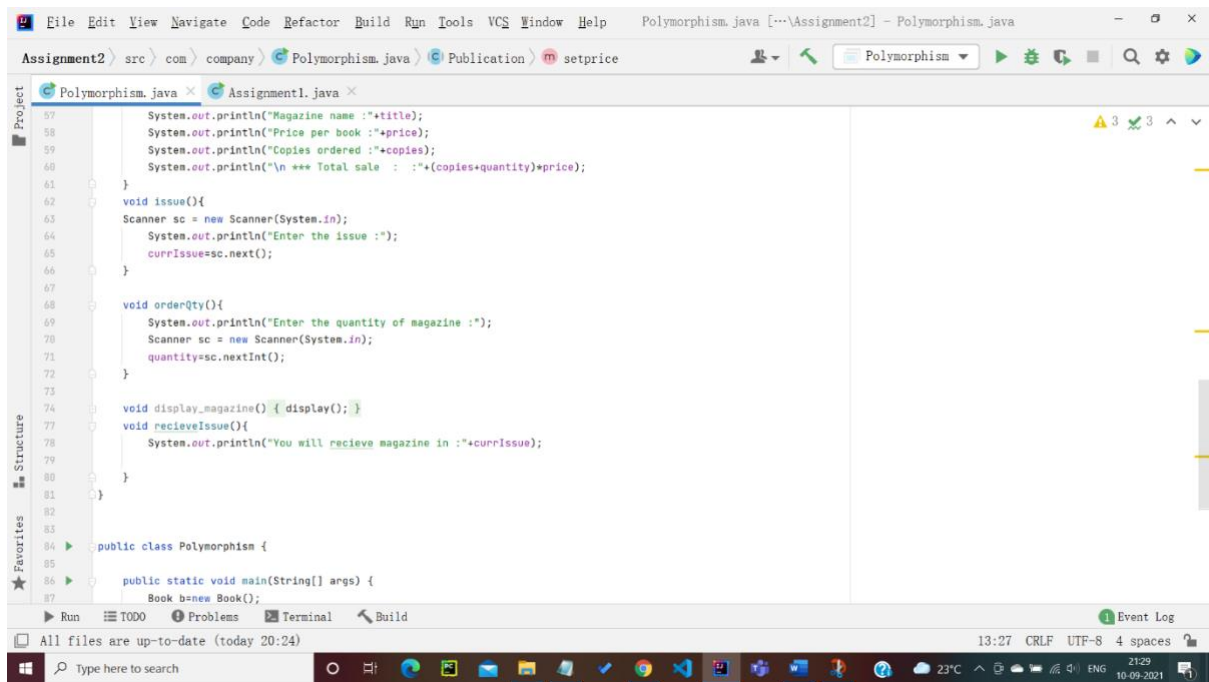
void setAuthor(){
    Scanner sc = new Scanner(System.in);
    System.out.print("Author Name :");
    author = sc.next();
}

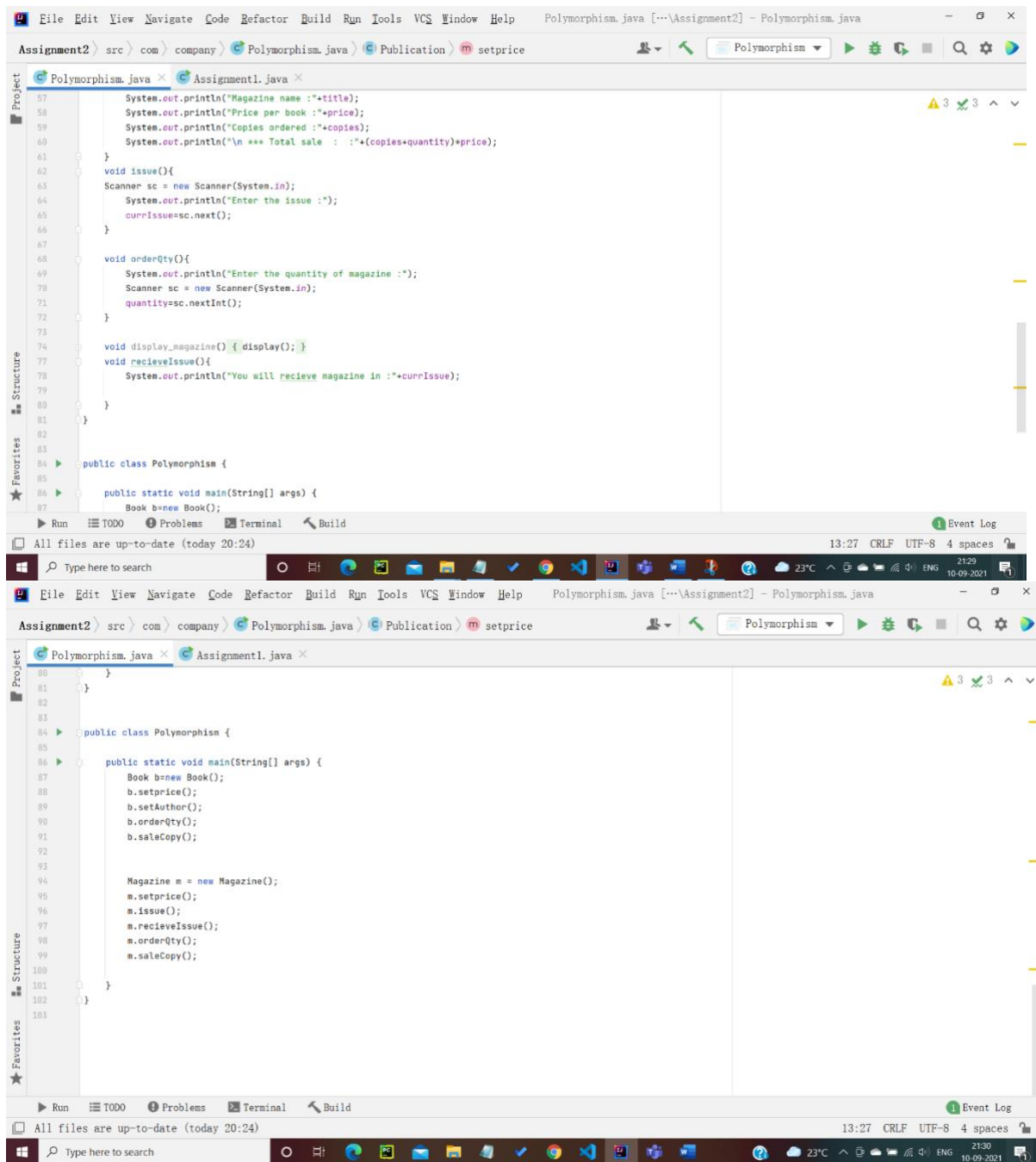
void orderQty(){
    System.out.println("Enter the quantity of the books :");
    Scanner sc=new Scanner(System.in);
    quantity=sc.nextInt();
}

void displayBook(){
    display();
    System.out.println("\t"+author);
}

}

class Magazine extends Publication{
    String curIssue;
    void saleCopy(){
        System.out.println("*****");
        System.out.println("Magazine name :"+title);
        System.out.println("Price per book :"+price);
    }
}
```





The screenshot shows the IntelliJ IDEA IDE with a project named "Assignment2". The "Run" window is open, displaying the output of a Java program. The program prompts the user to enter details for books and magazines, and then calculates the total sales for each.

```
"C:\Program Files\Java\jdk-10.0.2\bin\java.exe" "-javaagent:C:\Program Files\JetBrains\IntelliJ IDEA Community Edition 2021.2.1\lib\idea_rt.jar=54786:C:\Program Files\JetBrains\IntelliJ IDEA Community Edition 2021.2.1\bin"
Assignment2> src> com> company> Polymorphism.java> Publication> setprice
Polymorphism
Run: Polymorphism
Enter the title :Python
Enter the price :400
Enter the copies :70
Enter the quantity of the books :
70
*****
Book name :Python
Author name :Reena
Price per book :400
Copies ordered :70
*** Total sale : :56000
Enter the title :Java
Enter the price :450
Enter the copies :55
Enter the issue :
September
You will receive magazine in :September
Enter the quantity of magazine :
70
*****
Magazine name :Java
Price per book :450
Copies ordered :55
*** Total sale : :56250
Process finished with exit code 0
```

The bottom status bar shows "All files are up-to-date (a minute ago)", "31:1 CRLF UTF-8 4 spaces", and the system clock "21:31 10-09-2021".