Bridge course Assignment-5

Name:Balam Indira Priyadarsini

Date:30/06/25

Problem Solving Activity 1.1   :   Simple Dog class

1. Program Statement : we need to Create a class called Dog with: Class Attributes : species = "Canis familiaris ", numLegs = 4.And the List of the Instance Attributes :name, breed, age . And also need to Define a Method :bark() that prints "Woof!"

2. Algorithm:

Step 1:start program

Step 2: Creating a Dog class .Declare static class attributes: species, numLegs .

Step 3: Declare instance attributes: name, breed, age.

Step 4: Define a constructor to initialize instance attributes.

Step 5: Define a method bark() that prints "Woof!".

Step 6:In main, create dog objects and call bark() method.

Step 7:Print class attributes.

Step 8 :End of the program

3. Pseudocode:

Start

CLASS Dog

   STATIC ATTRIBUTE species = "Canis familiaris"

   STATIC ATTRIBUTE numLegs = 4

   INSTANCE ATTRIBUTES name, breed, age

   CONSTRUCTOR(name, breed, age):

SET instance variables

METHOD bark():

PRINT "Woof!"

IN MAIN:

CREATE dog1 = new Dog("Tommy", "Labrador", 5)

CALL dog1.bark()

CREATE dog2 = new Dog("Bruno", "Pug", 2)

CALL dog2.bark()

PRINT Dog.species

PRINT Dog.numLegs . End

4. Program Code:

```java
class Dog{
    static String species="Canis familiaris";
    static int numLegs=4;
    String name;
    String breed;
    int age;
    Dog(String name,String breed,int age){
        this.name=name;
        this.breed=breed;
        this.age=age;
    }
    void bark(){
        System.out.println(x:"Woof!");
    }
    public static void main(String[] args) {
        Dog dog1=new Dog(name:"Tommy",breed:"chichu",age:5);
        Dog dog2=new Dog(name:"Bruno",breed:"Pug",age:2);
        dog1.bark();
        dog2.bark();
        System.out.println("species:"+Dog.species);
        System.out.println("Number of legs:"+Dog.numLegs);
    }
}
```

5. Test Cases

Test Case No.	Input	Expected Output	Actual Output	Status (Pass/Fail)
1	dog1.bark()	Woof!	Woof!	Pass
2	dog1.bark()	Woof!	Woof!	Pass
3	Dog.species	Canis familiaris	Canis familiaris	Pass
4	Dog.numLegs	4	4	Pass

6. Output


```
PS C:\Users\Lenovo\OneDrive\Desktop\stemup\Assignment 5>
Woof!
Woof!
species:Canis familiaris
Number of legs:4
PS C:\Users\Lenovo\OneDrive\Desktop\stemup\Assignment 5>
```

7. Observation / Reflection

This program helped me understand Java class structure, the difference between static and instance attributes, and how to use constructors and methods. Though I faced minor confusion initially, practicing improved my clarity. It boosted my confidence in object-oriented programming and gave me a good foundation for building real-world Java programs.

Problem Solving Activity 1.2 : Book Class

1. Program Statement: Create a class named Book. And need to Add instance attributes: title (String,author (String),numPages (int),isOpen (boolean).Add two methods:openBook() is sets as isOpen to true, closeBook() is sets as isOpen to false

2. Algorithm:

Step 1:start program

Step 2:Create a class as a Book.

Step 3: Define instance variables: title, author, numPages, isOpen.

Step 4: Create a constructor to initialize title, author, numPages, and set isOpen to false by default

Step 5: Define openBook() to set isOpen = true

Step 6: Define closeBook() to set isOpen = false

Step 7: In the main method:

- Create an object of Book
- Call methods and check the value of isOpen

Step 8: End the program

3. Pseudocode

Start

CLASS Book:

    VARIABLES: title, author, numPages, isOpen

    METHOD Constructor(title, author, numPages):

        SET this.title = title

        SET this.author = author

        SET this.numPages = numPages

        SET isOpen = false

    METHOD openBook():

        SET isOpen = true

    METHOD closeBook():

        SET isOpen = false

    METHOD displayStatus():

        PRINT title, author, numPages, isOpen

MAIN:

    CREATE object book1 = new Book("Wings of Fire", "APJ Abdul Kalam", 180)

CALL book1.displayStatus()

CALL book1.openBook()

CALL book1.displayStatus()

CALL book1.closeBook()

CALL book1.displayStatus()

End

4. Program Code

```java
class Book {
    String title;
    String author;
    int numPages;
    boolean isOpen;
    Book(String title,String author,int numPages){
    this.title=title;
    this.author=author;
    this.numPages=numPages;
    this.isOpen=false;
}
void openBook(){
    isOpen=true;
}
void closeBook(){
    isOpen=false;
}
void displaystatus(){
    System.out.println("Book:"+title);
    System.out.println("Author"+author);
    System.out.println("Number of pages:"+numPages);
    System.out.println("Isopen?"+isOpen);
    System.out.println(x:"----------");
}
Run | Debug
public static void main(String[] args) {
    Book book1=new Book(title:"Wings of fire",author:"APJ Abdul Kalam", numPages:180);
    book1.displaystatus();
    book1.openBook();
    book1.displaystatus();
    book1.closeBook();
    book1.displaystatus();
}
}
```

5. Test Cases

Test Case No.	Input	Expected Output	Actual Output	Status (Pass/Fail)
1	Create book object	false	false	Pass
2	Call openBook()	true	true	Pass
3	Call closeBook()	false	false	Pass

6. Output

```
PS C:\Users\Lenovo\OneDrive\Desktop\stemup\Assignment 5> cd "c
Book:Wings of fire
AuthorAPJ Abdul Kalam
Number of pages:180
Isopen?false
---------
Book:Wings of fire
AuthorAPJ Abdul Kalam
Number of pages:180
Isopen?true
---------
Book:Wings of fire
AuthorAPJ Abdul Kalam
Number of pages:180
Isopen?false
---------
```

7. Observation / Reflection

This program helped me understand how to define a class with attributes and change its state using methods. I learned how to use a constructor to initialize values and how a method like openBook() can change the isOpen variable. It clearly shows how object properties can reflect real-world actions like opening and closing a book, which made learning object-oriented programming easier and more interesting.

Problem Solving Activity 1. 3 **: Identify Class Elements for Car Class**

1. Program Statement: For a class ,need to identify the 3-5 potential instance attribute and 2.3 potential instance methods.One appropriate class attribute shared by all Car objects.

2. Algorithm

Step 1:start program

Step 2: Define a class Car.Add a class attribute numWheels = 4.

Step 3: Add instance attributes: brand, model, year, color, isEngineOn.

Step 4: Create a constructor to initialize instance attributes, with isEngineOn set to false.

Step 5: Define the method startEngine() to set isEngineOn = true.

Step 6: Define the method stopEngine() to set isEngineOn = false.

Step 7: Define the method displayDetails() to print car info and status.

Step 8: In main(), create objects, call methods, and print results.

Step 9 :End the program

3. Pseudocode

Start

CLASS Car:

    CLASS ATTRIBUTE numWheels = 4

    INSTANCE ATTRIBUTES: brand, model, year, color, isEngineOn

    CONSTRUCTOR(brand, model, year, color):

        SET brand, model, year, color

        SET isEngineOn = false

    METHOD startEngine():

        SET isEngineOn = true

    METHOD stopEngine():

        SET isEngineOn = false

METHOD displayDetails():

   PRINT brand, model, year, color, isEngineOn

MAIN:

   CREATE car1 = new Car("Honda", "Civic", 2022, "Blue")
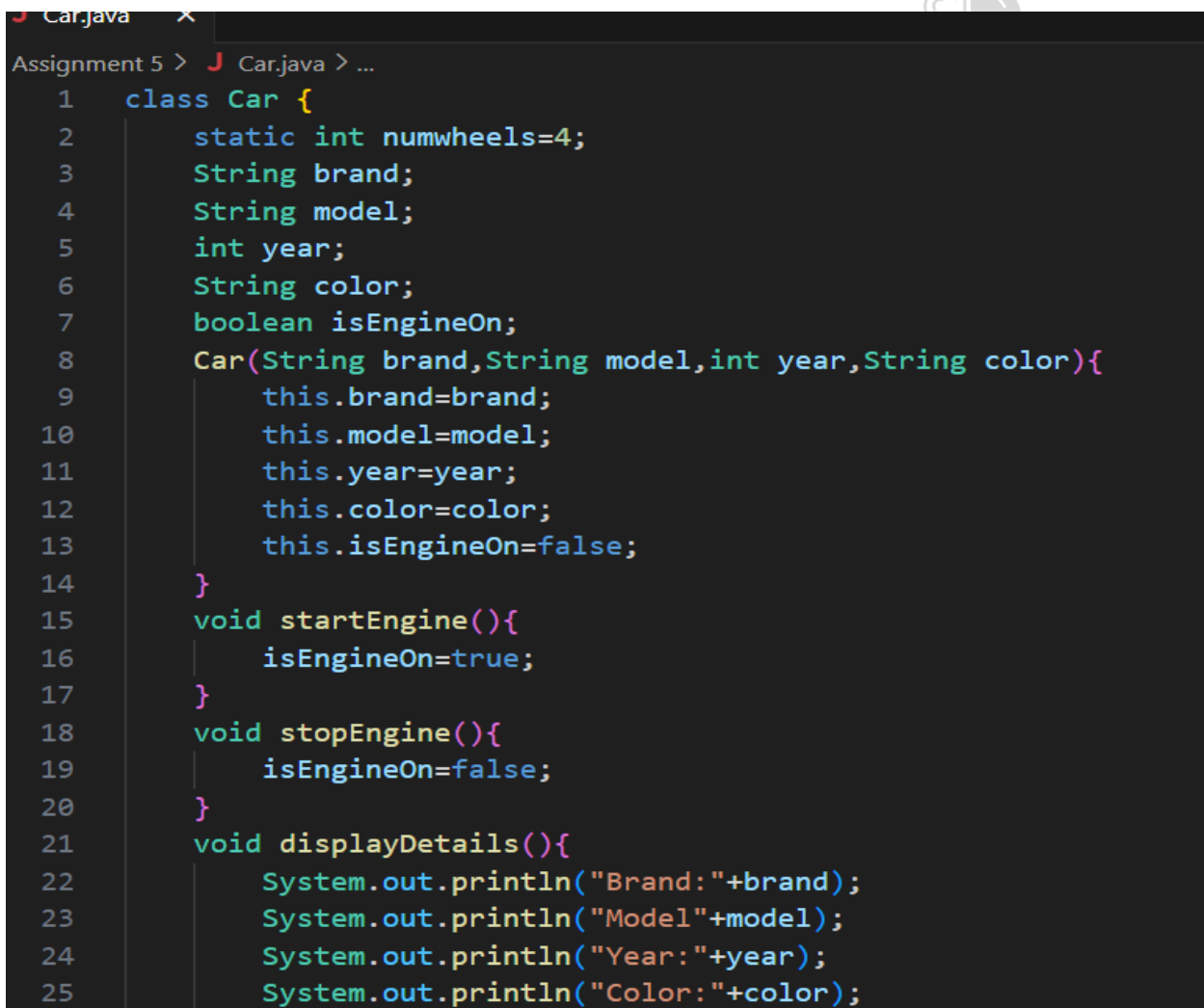
   CALL car1.displayDetails()

   CALL car1.startEngine()

   CALL car1.displayDetails()

   CALL car1.stopEngine()

   CALL car1.displayDetails()End

4. Program Code

```java
class Car {
    static int numwheels=4;
    String brand;
    String model;
    int year;
    String color;
    boolean isEngineOn;
    Car(String brand,String model,int year,String color){
        this.brand=brand;
        this.model=model;
        this.year=year;
        this.color=color;
        this.isEngineOn=false;
    }
    void startEngine(){
        isEngineOn=true;
    }
    void stopEngine(){
        isEngineOn=false;
    }
    void displayDetails(){
        System.out.println("Brand:"+brand);
        System.out.println("Model"+model);
        System.out.println("Year:"+year);
        System.out.println("Color:"+color);
```

```
21      void displayDetails(){
22          System.out.println("Brand:"+brand);
23          System.out.println("Model"+model);
24          System.out.println("Year:"+year);
25          System.out.println("Color:"+color);
26          System.out.println("Is Engine on?"+isEngineOn);
27          System.out.println(x:"Number of wheels:+numwheels");
28          System.out.println(x:"--------------------");
29
30      }
     Run | Debug
31      public static void main(String[] args) {
32          Car car1=new Car(brand:"Honda",model:"Civic",year:2025,color:"Blue");
33          Car car2=new Car(brand:"Toyota",model:"Corolla",year:2021,color:"white");
34          car1.displayDetails();
35          car1.startEngine();
36          car1.displayDetails();
37          car1.stopEngine();
38          car1.displayDetails();
39          car2.displayDetails();
40      }
41  }
42
```

5. Test Cases

Test Case No.	Input	Expected Output	Actual Output	Status (Pass/Fail)
1	Create Car("Honda", "Civic", 2022, "Blue")	isEngineOn = false	isEngineOn = false	pass
2	car1.startEngine()	isEngineOn = true	isEngineOn = true	Pass
3	car1.stopEngine()	isEngineOn = false	isEngineOn = false	pass

6. Output

```
PS C:\Users\Lenovo\OneDrive\Desktop\stemup\Assignment 5> cd "c:\Users\Lenov
Brand:Honda
ModelCivic
Year:2025
Color:Blue
Is Engine on?false
Number of wheels:+numwheels
-------------------
Brand:Honda
ModelCivic
Year:2025
Color:Blue
Is Engine on?true
Number of wheels:+numwheels
-------------------
Brand:Honda
ModelCivic
Year:2025
Color:Blue
Is Engine on?false
Number of wheels:+numwheels
-------------------
Brand:Toyota
ModelCorolla
Year:2021
Color:white
Is Engine on?false
Number of wheels:+numwheels
-------------------
```

7. Observation / Reflection

This program helped me understand how to design a class with both shared (class) and unique (instance) properties. The use of the numWheels class attribute showed me how data can be shared across all objects, while methods like startEngine() and stopEngine() helped me practice modifying object state. Initially, I was confused about static vs non-static, but through testing and practice, I understood their differences.

Problem Solving Activity 2. 1   :   Create Dogs

1. Program Statement: Create Dogs as a class and to create 2 Dog objects:

- Dog1:"Buddy","Golden Retriever",5
- Dog2:"Lucky","Poodle",2
  Call the bark() method and print names and ages

2. Algorithm

Step 1:start program

Step 2: Create a class named Dog.

Step 3: Define attributes: name, breed, age.

Step 4: Add a constructor to initialize these attributes. Add a method bark() to display a barking message.

Step 6: Create 2 Dog objects with the given data.Call bark() method for each dog.print their name and age.

Step 7 :End the program

3. Pseudocode

Start

Class Dog:

   Attributes:name ,breed, age

   Constructor(name, breed, age):

      set the attributes

   Method bark():

      print "name says Woof!"

Main:

   Create Dog1 as Dog("Buddy", "Golden Retriever", 5)

   Create Dog2 as Dog("Lucky", "Poodle", 2)

   Call Dog1.bark()

   Print Dog1 name and age

Call Dog2.bark()

Print Dog2 name and age

End

4. Program Code

```java
public class CreateDog {
    String name;
    String breed;
    int age;
    public CreateDog(String name,String breed,int age){
        this.name=name;
        this.breed=breed;
        this.age=age;
    }
    public void bark(){
        System.out.println(name+"Says Woof!");
    }
    Run | Debug
    public static void main(String[] args) {
        CreateDog dog1=new CreateDog(name:"Buddy",breed:"Golden Retriever",age:5);
        dog1.bark();
        System.out.println("Name"+dog1.name+",Age:"+dog1.age);

        CreateDog dog2=new CreateDog(name:"Lucky",breed:"Poodle",age:2);
        dog2.bark();
        System.out.println("Name:"+dog2.name+",Age:"+dog2.age);
    }
}
```

5. Test Cases

Test Case No.	Input	Expected Output	Actual Output	Status (Pass/Fail)
1	Dog("Buddy", "Golden Retriever", 5)	Buddy says Woof! Name: Buddy, Age: 5	Buddy says Woof! Name: Buddy, Age: 5	Pass
2	Dog("Lucky", "Poodle", 2)	Lucky says Woof! Name: Lucky, Age: 2	Lucky says Woof! Name: Lucky, Age: 2	Pass

6. Output


```
PS C:\Users\Lenovo\OneDrive\Desktop\stemup\Assignment 5> cd "c:\
ava CreateDog }
BuddySays Woof!
NameBuddy,Age:5
LuckySays Woof!
Name:Lucky,Age:2
```

7. Observation / Reflection

While working on this Java program to create a Dog class and two objects, I understood key object-oriented concepts like class structure, constructors, and method usage. I initially struggled with using constructors and assigning values correctly using the this keyword, but resolved it through practice. Creating the bark() method showed how object behavior can be defined and reused. I also learned to access object data using dot notation and improved how I structured output.

Problem Solving Activity 2.2   : Manage Books

1. Program Statement:

Create two Book objects with constructor and Implement displayStatus()

Void displayStatus(){

String status=isOpen? "Open":"Closed";

System.out.println(title+"by"+author+"is"+status);

}

2. Algorithm

Step 1:start program

Step 2: Create a Book class with attributes: title, author, isOpen

Step 3: Add a constructor to initialize the values.

Step 4: Create a method displayStatus() that:

- Checks isOpen.
- If true, print "Open", else print "Closed".

Step 5: In main(), create two book objects with different values.Call displayStatus() for each object.

Step 6:End the program.

3. Pseudocode

Start

Class Book:

   Attributes: title, author, isOpen

   Constructor(title, author, isOpen):

      set the attributes

   Method displayStatus():

      if isOpen is true:

         status = "Open"

      else:

         status = "Closed"

      print title + " by " + author + " is " + status

Main:

   Create Book1 with ("Java Basics", "James", true)

   Create Book2 with ("Python Intro", "Guido", false)

   Call Book1.displayStatus()
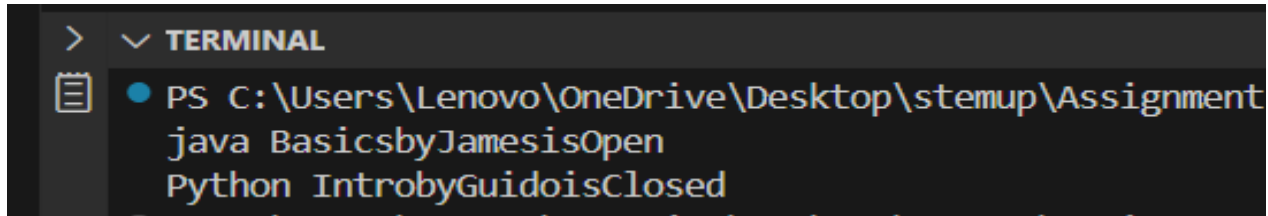
   Call Book2.displayStatus()End

4. Program Code

```java
public class ManageBooks {
static class Book{
    String title;
    String author;
    boolean isOpen;
    public Book(String title,String author,boolean isOpen){
        this.title=title;
        this.author=author;
        this.isOpen=isOpen;
    }
    public void displayStatus(){
        String status;
        if(isOpen){
            status="Open";
        }else{
            status="Closed";
        }
        System.out.println(title +"by"+author+"is"+status);
    }
}
    Run | Debug
    public static void main(String[] args) {
        Book book1=new Book(title:"java Basics",author:"James",isOpen:true);
        Book book2=new Book(title:"Python Intro", author:"Guido", isOpen:false);
        book1.displayStatus();
        book2.displayStatus();
    }
}
```

5. Test Cases

Test Case No.	Input	Expected Output	Actual Output	Status (Pass/Fail)
1	Book("Java Basics", "James", true)	Java Basics by James is Open	Java Basics by James is Open	Pass
2	Book("Python Intro", "Guido", false)	Python Intro by Guido is Closed	Python Intro by Guido is Closed	Pass

6. Output


```
TERMINAL
PS C:\Users\Lenovo\OneDrive\Desktop\stemup\Assignment
java BasicsbyJamesisOpen
Python IntrobyGuidoisClosed
```

7. Observation / Reflection

While developing this Java program to manage book objects, I learned how to use constructors to initialize object properties and apply conditional logic using the ternary operator within a method. Initially, I faced challenges in handling boolean values correctly and printing formatted output, especially when combining strings and variables. Implementing the displayStatus() method showed how object behavior can dynamically change based on internal state (isOpen). This helped reinforce how to use class attributes, constructors for initialization, and method logic for decision-making.

Problem Solving Activity 2.3   :   Student Record

1. Program Statement:

For a given program need to record the student Records

Public class Student{

String name;

String idNumber;

String major;

Public Student(String name,String idNumber,String major){
this.name=name;

this.idNumber=idNumber;

this.major=major;

}

Public String gerInfo(){
return name+",ID:"+idNumber+",Major:"+major;

}

}

Create 3-Student objects and print their info

2. Algorithm

Step 1:start program

Step 2:Define the Student class with 3 attributes.

Step 3: Create a constructor to initialize those attributes.

Step 4: Write a method getInfo() to return a string with student details.

Step 5: In main(), create 3 objects of Student with different data.

Step 6:End the program.

3. Pseudocode

start

Class Student:

  Attributes: name, idNumber, major

  Constructor(name, idNumber, major):

    Set values to instance variables

  Method getInfo():

    Return formatted string: name, ID: idNumber, Major: major

Main:

  Create student1 with ("Alice", "S001", "Computer Science")

  Create student2 with ("Bob", "S002", "Electronics")

  Create student3 with ("Charlie", "S003", "Mechanical")

  Print student1.getInfo()

  Print student2.getInfo()

  Print student3.getInfo()

END

4. Program Code

```java
Assignment 5 > J StudentRecord.java > StudentRecord > Student > getInfo()
1   public class StudentRecord {
2       static class Student{
3           String name;
4           String idNumber;
5           String major;
6           public Student(String name,String idNumber,String major){
7               this.name=name;
8               this.idNumber=idNumber;
9               this.major=major;
10          }
11          public String getInfo(){
12              return name+ ",ID:"+idNumber+ ",Major:"+major;
13
14          }
15      }
    Run | Debug
16      public static void main(String[] args) {
17          Student student1=new Student(name:"Alice",idNumber:"S001",major:"computer Science");
18          Student student2=new Student(name:"Bob",idNumber:"S002",major:"Electronics");
19          Student student3=new Student(name:"Charlie",idNumber:"S0013",major:"Mechanical");
20          System.out.println(student1.getInfo());
21          System.out.println(student2.getInfo());
22          System.out.println(student3.getInfo());
23      }
24  }
```

5. Test Cases

Test Case No.	Input	Expected Output	Actual Output	Status (Pass/Fail)
1	("Alice", "S001", "Computer Science")	Alice, ID: S001, Major: Computer Science	Alice, ID: S001, Major: Computer Science	Pass
2	("Bob", "S002", "Electronics")	Bob, ID: S002, Major: Electronics	Bob, ID: S002, Major: Electronics	Pass
3	("Charlie", "S003", "Mechanical")	Charlie, ID: S003, Major: Mechanical	Charlie, ID: S003, Major: Mechanical	Pass

6. Output

```
\Desktop\Stemup\Assignment 5\    ; if ($?) { javac Studen
Alice,ID:S001,Major:computer Science
Bob,ID:S002,Major:Electronics
Charlie,ID:S0013,Major:Mechanical
PS C:\Users\Lenovo\OneDrive\Desktop\stemup\Assignment
```

7. Observation / Reflection

While building this program, I clearly understood how to define a class with multiple attributes, initialize them using a constructor, and access the data using a method. Initially, I had trouble with proper syntax for method return types and string formatting, but after correcting it, I learned how useful methods like getInfo() are in organizing code. Creating and printing multiple student records helped reinforce object creation, constructor usage, and data access through methods.

Problem Solving Activity 3.1   :Bank Account

1. Program Statement:

Create and test a BankAccount as a class with getBalance(),deposit(),and withdraw() as a methods.

And also Use Try for invalid operations(e.g.,negative deposit,excessive withdrawal)

2. Algorithm

Step 1:start program

Step 2: Create a class BankAccount with a balance variable and a constructor to initialize the balance.

Step 3: Define getBalance() to return current balance.

Step 4: Define deposit(amount):

- if amount >balance,print "Insufficient balance".
- If amount <0,print "Invalid amount".
- Else,subtract from balance.

Step 6: In main(), create BankAmount object. Test deposit,withdraw,and display operations.

Step 5: End the program

3. Pseudocode

Start

Class BankAccount:

   Attribute: balance

   Constructor(initialBalance):

     set balance

Method getBalance():

   return balance

Method deposit(amount):

   if amount > 0:

      balance += amount

   else:

      print "Invalid deposit amount"

Method withdraw(amount):

   if amount <= 0:

      print "Invalid withdrawal amount"

   else if amount > balance:

      print "Insufficient balance"

   else:

      balance -= amount

Main:

Create account with initial balance

account.deposit(1000)

account.withdraw(500)

account.withdraw(700)

account.deposit(-100)

Print account.getBalance()End

4. Program Code

```java
public class BankApp {
    static class BankAccount {
        private double balance;

    public BankAccount(double initialBalance){
        this.balance=initialBalance;
    }
    public double getBalance(){
        return balance;
    }
    public void deposit(double amount){
        if(amount>0){
            balance +=amount;
            System.out.println("Deposited:"+amount);
        }else{
            System.out.println(x:"Invalid deposit amount!");
        }
    }
    public void withdraw(double amount){
        if(amount<=0){
            System.out.println(x:"Invalid withdrawal amount!");
        }else if(amount >balance){
            System.out.println(x:"Insufficient blance!");
        }else{
            balance-=amount;
            System.out.println("Withdrawn:"+amount);
```

```
26                    System.out.println("Withdrawn:"+amount);
27            }
28        }
29    }
      Run | Debug
30    public static void main(String[] args){
31        BankAccount account=new BankAccount(initialBalance:1000.0);
32        account.deposit(amount:500.0);
33        account.withdraw(amount:300.0);
34        account.withdraw(amount:1500.0);
35        account.deposit(-100.0);
36        account.withdraw(-50);
37        System.out.println("Final BAlance:"+account.getBalance());
38    }
39    }
```

5. Test Cases

Test Case No.	Input	Expected Output	Actual Output	Status (Pass/Fail)
1	Deposit: 500	Deposited: 500	Deposited: 500	Pass
2	Withdrawn: 300	Withdrawn: 300	Withdrawn: 300	Pass
3	Withdraw (Excess) 1500	Insufficient balance	Insufficient balance	pass

6. Output

```
Drive\Desktop\stemup\Assignment 5> cd "c:\Users\Leno
Deposited:500.0
Withdrawn:300.0
Withdrawn:300.0
Insufficient blance!
Invalid deposit amount!
Invalid withdrawal amount!
Final BAlance:1200.0
```

7. Observation / Reflection

While working on the BankAccount Java program, I gained practical experience in handling real-world operations like deposit and withdrawal with proper validations. I understood how to use

condition checks to manage invalid inputs such as negative deposits or withdrawals beyond the current balance. Initially, I was confused about where to place validations, but implementing them inside the method helped organize the logic better. This program also helped reinforce the importance of data encapsulation using private variables and accessing them via public methods.

Problem Solving Activity 3.2 : Product Inventory

1. Program Statement:

With the use of below program need to form the product inventory as

Public class Product{

Private String name;

Private double price;

Private int quantity;

Public product(String name,double price,int quantity){

this.name=name;

setPrice(price);

setQuantity(quantity);

}

Public String getName(){return name;}

Public double getPrice() { return price; }

Public int getQuantity() { return quantity;}

Public void setPrice (double price){

If(price >0){

this.price=price;

}

}

Public double getTotalValue(){

Return price*quantity;

}

}

As Test object creation ,use getters/setters, validate constraints

2. Algorithm

Step 1:start program

Step 2: Create a Product class and private attributes:name,price,quantity

Step 3: Use setprice() and setquantity() in the constructor to validate values.

Step 4: Create getter methods to return name, price, and quantity.To define getTotalValue() to return price × quantity.

Step 5:In main(),create product objects and use getters to display product info.invalid prices and verify they are handled.

Step 6: End the program

3. Pseudocode

Start

Class Product:

  Attributes: name, price, quantity

  Constructor(name, price, quantity):

    set name

    set price via setprice()

    set quantity via setquantity()

  Getter methods: getName(), getprice(), getquantity()

  Setter method: setprice(price) should only if price > 0

  Method: getTotalValue() = price × quantity

Main:

  Create product1 ("Laptop", 50000, 2)

  Create product2 ("Phone", -10000, 5)it should not set negative price

  Display name, price, quantity, and total value for each

End

4. Program Code

```java
public class ProductInventory {
    static class Product {
        private String name;
        private double price;
        private int quantity;
        public Product(String name,double price,int quantity){
            this.name=name;
            setprice(price);
            setquantity(quantity);
        }
        public String getName(){
            return name;
        }
        public double getprice(){
            return price;
        }
        public int getquantity(){
            return quantity;
        }
        public void setprice(double price){
            if(price>0){
                this.price=price;
            }else{
                System.out.println(x:"Invalid quantity!Must be non-negative");
            }
        }
        public void setquantity(int quantity){
            if(quantity >=0){
                this.quantity=quantity;
            }else{
                System.out.println(x:"Invalid quantity! Must be non-negative.");
            }
```

```java
        }else{
            System.out.println(x:"Invalid quantity! Must be non-negative.");
        }
    }
    public double getTotalValue(){
        return price*quantity;
    }
}

public static void main(String[] args) {
    Product p1=new Product(name:"Laptop", price:50000, quantity:2);
    Product p2 = new Product(name:"Phone", price:10000, quantity:5);
    Product p3 = new Product(name:"Tablet", price:15000, quantity:3);
    System.out.println(p1.getName()+":Price="+p1.getprice()+",Quantity="+p1.getquantity()+",Total Value="+p1.getTotalValue());
    System.out.println(p2.getName()+":Price="+p2.getprice()+ ", Quantity = " + p2.getquantity() + ", Total Value = " + p2.getTotalValue());
    System.out.println(p3.getName()+":Price="+p3.getprice()+", Quantity = " + p3.getquantity() + ", Total Value = " + p3.getTotalValue());
}
}
```

5. Test Cases

Test Case No.	Input	Expected Output	Actual Output	Status (Pass/Fail)
1	Laptop,50000,2	Valid: Total Value = 100000	Valid: Total Value = 100000	Pass
2	Phone,-10000,5	Invalid price =should not update	Invalid price =should not update	Pass
3	Tablet,15000,3	Valid: Total Value = 45000	Valid: Total Value = 45000	pass

6. Output

```
Laptop:Price=50000.0,Quantity=2,Total Value=100000.0
Phone:Price=10000.0, Quantity = 5, Total Value = 50000.0
Tablet:Price=15000.0, Quantity = 3, Total Value = 45000.0
PS C:\Users\Lenovo\OneDrive\Desktop\stemup\Assignment 5> 
```

7. Observation / Reflection

In this Product Inventory program, I learned how to encapsulate data using private variables and use getters and setters to access and modify them securely. The validation in the setprice() and setquantity() methods helped me understand how to prevent invalid data from entering the object. Initially, I forgot to use the setter in the constructor, which caused unvalidated values to be accepted, but fixing it taught me the importance of centralized validation logic. Creating multiple product objects and checking outputs gave me confidence in implementing real-world logic with conditions.