

Learner Assignment Submission Format

Learner Details

- **Name:** Adnan Sameer
 - **Enrollment Number:**
 - **Batch / Class:**
 - **Assignment:** (Bridge Course Day 4)
 - **Date of Submission:** 30-06-2025
-

Problem Solving Activity 5.1

1. Real-World Object Dissection Identify three real-world objects and describe:

- At least 3-5 attributes
 - 2-3 behaviors
-

2. Algorithm

- Define a class named Dog with static and instance variables.
 - Initialize static variables species and numLegs (shared across all Dog objects).
 - Define instance variables name, breed, and age.
 - Create a constructor to initialize name, breed, and age.
 - Define a method bark() to print a dog's barking sound.
 - In the main method:
 - Create a Dog object named myDog.
 - Display the dog's name, breed, age, species, and number of legs.
 - Call the bark() method.
-

3. Pseudocode

CLASS Dog

 STATIC VARIABLE species = "Canis familiaris"

 STATIC VARIABLE numLegs = 4

INSTANCE VARIABLES

name

breed

age

CONSTRUCTOR Dog(name, breed, age)

SET this.name = name

SET this.breed = breed

SET this.age = age

METHOD bark()

PRINT "Woof!"

MAIN METHOD

CREATE Dog object myDog with name "Buddy", breed "Labrador", age 3

PRINT myDog.name

PRINT myDog.breed

PRINT myDog.age

PRINT Dog.species

PRINT Dog.numLegs

CALL myDog.bark()

4. Program Code

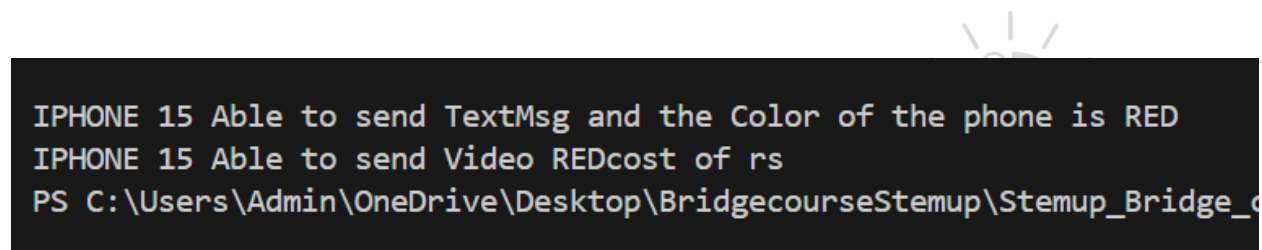
```
public class D5_1ApplePhone {
    String model="IPHONE 15";
    double cost=69000;
    String color=" RED";
    public void makeCall(){
        System.out.println(model+" Make CaLL");
    }
    public void SendTextMsg(){
        System.out.println(model+" Able to send TextMsg and the "+"Color of the
phone is"+color);
    }
    public void SendVideo(){
```

```

        System.out.println(model+" Able to send Video"+ color +"cost of rs");
    }
    public static void main(String[] args) {
        D5_1ApplePhone b=new D5_1ApplePhone();
        b.makeCall();
        b.SendTextMsg();
        b.SendVideo();
    }
}

```

6. Screenshots of Output



```

IPHONE 15 Able to send TextMsg and the Color of the phone is RED
IPHONE 15 Able to send Video REDcost of rs
PS C:\Users\Admin\OneDrive\Desktop\BridgecourseStemup\Stemup_Bridge_c

```

7. Observation / Reflection

By creating an object. We are calling the method .previously we are calling method by method name .now we are calling it by creating an object b

Problem Solving Activity 5.2

1. Program for Escalator

2. Algorithm

- ☐ Define a class named D5_2Escalator.
- ☐ Initialize instance variables:
 - cost with value 69000
 - type1 with value "Belt-type escalators"
 - type2 with value "Stairway escalators"
- ☐ Create method CarryPeoplesGoods():

- Print message that type1 escalators can carry people and goods, and their cost.
- ☐ Create method CarryPeoples():
 - Print message that type2 escalators carry only people, and their cost.
- ☐ Create method MoveingDirection():
 - Print that both type1 and type2 can move in both directions.
- ☐ In the main() method:
 - Create an object of D5_2Escalator
 - Call all three methods using the object.

3. Pseudocode

CLASS D5_2Escalator

VARIABLE cost = 69000

VARIABLE type1 = "Belt-type escalators"

VARIABLE type2 = "Stairway escalators"

METHOD CarryPeoplesGoods()

PRINT type1 + " Can Carry peoples, goods and the Cost is " + cost

METHOD CarryPeoples()

PRINT type2 + " Can Carry only peoples not goods and the Cost is " + cost

METHOD MoveingDirection()

PRINT type1 + " AND " + type2 + " Able to Move front and back"

MAIN METHOD

CREATE object b of class D5_2Escalator

CALL b.CarryPeoplesGoods()

CALL b.CarryPeoples()

CALL b.MoveingDirection()

4. Program Code

```
public class D5_2Escalator {
    double cost=69000;
    String type1="Belt-type escalators";
    String type2="Stairway escalators";
    public void CarryPeoplesGoods(){
        System.out.println(type1+"Can Carry peoples, goods and the Cost is "+cost);
    }
    public void CarryPeoples(){
        System.out.println(type2+"Can Carry only peoples not goods and the Cost is "+cost);
    }
    public void MoveingDirection(){
        System.out.println(type1+" AND "+type2+" Able to Move front and back");
    }
    public static void main(String[] args) {
        D5_2Escalator b=new D5_2Escalator();
        b.CarryPeoplesGoods();
        b.CarryPeoples();
        b.MoveingDirection();
    }
}
```

5. Observation / Reflection

By creating an object. We are calling the method .previously we are calling method by method name .now we are calling it by creating an object b

Problem Solving Activity 5.3

1. Program for Bike

2. Algorithm

1. Define a class D5_3Bikes with attributes: cost, type, and Color.
 2. Define methods:
 - Speed(): Print the speed of the bike based on its type.
 - millage(): Print the mileage of the bike based on its type.
 - Looks(): Print the cost and color of the bike.
 3. In the main method:
 - Create an instance of the D5_3Bikes class.
 - Call the Speed(), millage(), and Looks() methods.
-

3. Pseudocode

CLASS D5_3Bikes

ATTRIBUTES

cost = 100000

type = "Petrol"

Color = "Black"

METHOD Speed()

PRINT type + " engine Speed is some speed"

METHOD millage()

PRINT type + " engine " + " millage is -----"

METHOD Looks()

PRINT "The cost of bike is " + cost + " and the color is " + Color

MAIN

CREATE INSTANCE b OF D5_3Bikes

CALL b.Speed()

CALL b.millage()

CALL b.Looks()

4. Program Code

```
public class D5_3Bikes {
    double cost=100000;
    String type="Petrol";
    String Color="Black";
    public void Speed(){
        System.out.println(type+" engine Speed is some speed");
    }
    public void millage(){
        System.out.println(type+" engine "+" millage is -----");
    }
    public void Looks(){
        System.out.println(" The cost of bike is "+cost+" and the color is "+Color);
    }
    public static void main(String[] args) {
        D5_3Bikes b=new D5_3Bikes();
        b.Speed();
        b.millage();
        b.Looks();
    }
}
```

```
Petrol engine Speed is some speed
Petrol engine millage is -----
The cost of bike is 100000.0 and the color is Black
PS C:\Users\Admin\OneDrive\Desktop\BridgecourseStemup\St
```

7. Observation / Reflection

By creating an object. We are calling the method .previously we are calling method by method name .now we are calling it by creating an object b

Problem Solving Activity 1.2

1. Procedural vs. Object-Oriented Thought Write how you'd model a list of customers:

Procedurally using arrays and methods

OOP using a Customer class with attributes (name, id) and behaviors (addCustomer(), deleteCustomer())

2. Algorithm

1. Define a class D5_5 with static arrays names and ids to store customer data, and a static variable count to track the number of customers.
 2. Create a method addCustomer(name, id) that:
 - Adds a new customer to the names and ids arrays.
 - Increments the count variable.
 3. Create a method showCustomers() that:
 - Iterates through the names and ids arrays up to the count index.
 - Prints each customer's ID and name.
 4. In the main method:
 - Call addCustomer() to add customers to the system.
 - Call showCustomers() to display the added customers.
-

3. Pseudocode

CLASS D5_5

STATIC ARRAYS

names[10]

ids[10]

STATIC VARIABLE

```
count = 0
```

METHOD addCustomer(name, id)

```
names[count] = name
```

```
ids[count] = id
```

```
count = count + 1
```

METHOD showCustomers()

```
FOR i FROM 0 TO count - 1
```

```
PRINT ids[i] + " - " + names[i]
```

MAIN

```
CALL addCustomer("Alice", 1)
```

```
CALL addCustomer("Bob", 2)
```

```
CALL showCustomers()
```

4. Program Code

```
public class D5_5 {
    static String[] names = new String[10];
    static int[] ids = new int[10];
    static int count = 0;

    public static void addCustomer(String name, int id) {
        names[count] = name;
        ids[count] = id;
        count++;
    }

    public static void showCustomers() {
        for (int i = 0; i < count; i++) {
            System.out.println(ids[i] + " - " + names[i]);
        }
    }

    public static void main(String[] args) {
        addCustomer("Alice", 1);
    }
}
```

```

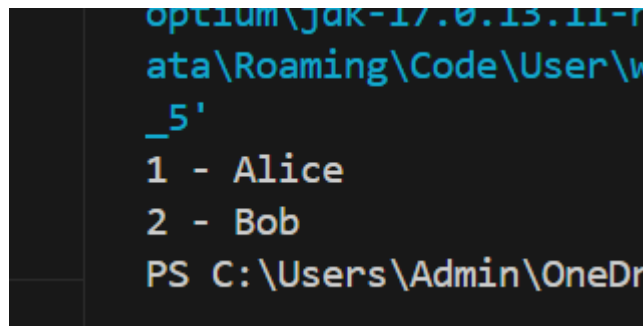
        addCustomer("Bob", 2);
        showCustomers();
    }
}

```

5. Test Cases

Test Case No.	Input	Expected Output	Actual Output	Status (Pass/Fail)
1	Alice-1	Alice-1	Alice-1	pass
2	Bob-2	Bob-2	Bob-2	pass

6. Screenshots of Output



```

optium\jdk-17.0.13.11-n
ata\Roaming\Code\User\w
_5'
1 - Alice
2 - Bob
PS C:\Users\Admin\OneDr

```

7. Observation / Reflection

The code effectively stores and retrieves customer data using arrays.

Problem Solving Problem 1.1

1. Create simple dog class


Define a class named Dog:

Add class attributes: species = "Canis familiaris" and numLegs = 4

List the structure of instance attributes: name, breed, age

Define a method bark() that prints "Woof!"

2. Algorithm

1. Define a class Dog with static attributes species and numLegs, and instance attributes name, breed, and age.
 2. Create a constructor Dog that initializes name, breed, and age.
 3. Define a method bark() that prints "Woof!".
 4. In the main method:
 - Create a new Dog object with specified name, breed, and age.
 - Print the dog's details: name, breed, age, species, and numLegs.
 - Call the bark() method.
- 
-

3. Pseudocode

CLASS Dog

STATIC ATTRIBUTES

species = "Canis familiaris"

numLegs = 4

ATTRIBUTES

name

breed

age

CONSTRUCTOR Dog(name, breed, age)

 INITIALIZE name, breed, age

METHOD bark()

 PRINT "Woof!"

MAIN

 CREATE Dog myDog WITH name = "Buddy", breed = "Labrador", age = 3

 PRINT "Name: " + myDog.name

 PRINT "Breed: " + myDog.breed

 PRINT "Age: " + myDog.age

 PRINT "Species: " + Dog.species

 PRINT "Number of Legs: " + Dog.numLegs

 CALL myDog.bark()



4. Program Code

```
public class Dog {
    static String species = "Canis familiaris";
    static int numLegs = 4;

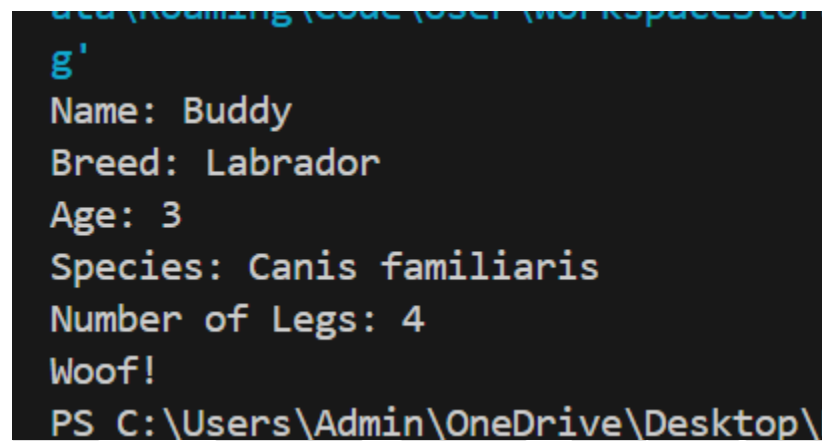
    String name;
    String breed;
    int age;
    public Dog(String name, String breed, int age) {
        this.name = name;
        this.breed = breed;
        this.age = age;
    }

    public void bark() {
        System.out.println("Woof!");
    }

    // Main method to test the Dog class
    public static void main(String[] args) {
        Dog myDog = new Dog("Buddy", "Labrador", 3);
        System.out.println("Name: " + myDog.name);
    }
}
```

```
        System.out.println("Breed: " + myDog.breed);  
        System.out.println("Age: " + myDog.age);  
        System.out.println("Species: " + Dog.species);  
        System.out.println("Number of Legs: " + Dog.numLegs);  
        myDog.bark();  
    }  
}
```

5. Screenshots of Output



```
g'  
Name: Buddy  
Breed: Labrador  
Age: 3  
Species: Canis familiaris  
Number of Legs: 4  
Woof!  
PS C:\Users\Admin\OneDrive\Desktop\B
```

6. Observation / Reflection

Problem Solving Problem 1.2

1. Basic Book Class

Define a class named Book:

Add instance attributes: title, author, numPages, isOpen

Add two methods: openBook() (sets isOpen to true) and closeBook() (sets isOpen to false)

2. Algorithm

1. Define a class D5Book with attributes: title, author, numPages, and isOpen.
2. Create a constructor D5Book that initializes title, author, numPages, and sets isOpen to true.
3. Define methods:
 - openBook(): Sets isOpen to true and prints a confirmation message.
 - closeBook(): Sets isOpen to false and prints a confirmation message.
 - displayInfo(): Prints the book's details: title, author, numPages, and isOpen.
4. In the main method:
 - Create a new D5Book object with specified title, author, and numPages.
 - Call displayInfo() to show the book's initial state.
 - Call openBook() and closeBook() to demonstrate book state changes.
 - Call displayInfo() again to show the final state.

3. Pseudocode

CLASS D5Book

ATTRIBUTES

title

author

numPages

isOpen

CONSTRUCTOR D5Book(title, author, numPages)

INITIALIZE title, author, numPages

SET isOpen TO true

METHOD openBook()

SET isOpen TO true

PRINT "The book is now open."

METHOD closeBook()

SET isOpen TO false

```
PRINT "The book is now closed."
```

```
METHOD displayInfo()
```

```
PRINT "Title: " + title
```

```
PRINT "Author: " + author
```

```
PRINT "Number of Pages: " + numPages
```

```
PRINT "Is the book open? " + isOpen
```

```
MAIN
```

```
CREATE D5Book myBook WITH title = "MY BOOK", author = "Sameer", numPages = 100
```

```
CALL myBook.displayInfo()
```

```
CALL myBook.openBook()
```

```
CALL myBook.closeBook()
```

```
CALL myBook.displayInfo()
```



4. Program Code

```
public class D5Book {
    String title;
    String author;
    int numPages;
    boolean isOpen;

    public D5Book(String title, String author, int numPages){
        this.title = title;
        this.author = author;
        this.numPages = numPages;
        this.isOpen = true;
    }

    public void openBook() {
        isOpen = true;
        System.out.println("The book is now open.");
    }

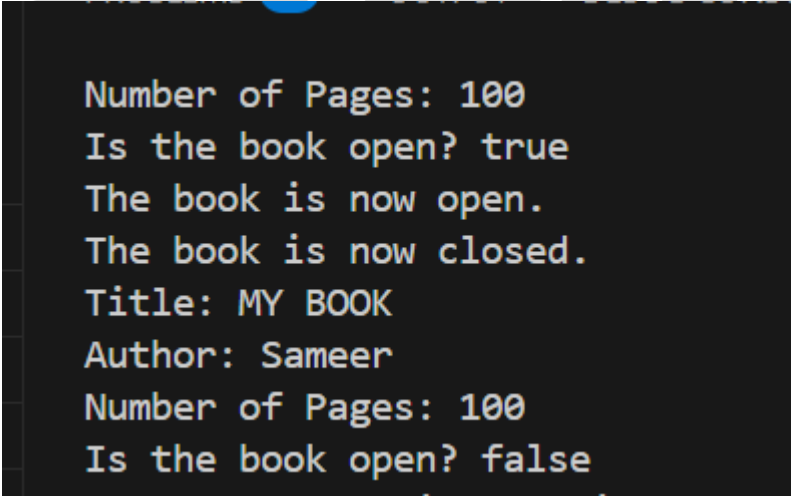
    public void closeBook() {
```

```
        isOpen = false;
        System.out.println("The book is now closed.");
    }

    public void displayInfo() {
        System.out.println("Title: " + title);
        System.out.println("Author: " + author);
        System.out.println("Number of Pages: " + numPages);
        System.out.println("Is the book open? " +isOpen);
    }

    public static void main(String[] args) {
        D5Book myBook = new D5Book("MY BOOK", "Sameer", 100);
        myBook.displayInfo();
        myBook.openBook();
        myBook.closeBook();
        myBook.displayInfo();
    }
}
```

6. Screenshots of Output



```
Number of Pages: 100
Is the book open? true
The book is now open.
The book is now closed.
Title: MY BOOK
Author: Sameer
Number of Pages: 100
Is the book open? false
```

7. Observation / Reflection

Problem Solving Problem 1.3

1. Identify Class Elements for Car Class

For a Car class, identify:

3-5 potential instance attributes

2-3 potential instance methods

2. Algorithm

- Define a class Car with the following attributes:

- model_name (String)
- cost (int)
- type (String, e.g., petrol/diesel/electric)
- color (String)

- ☐ Create a **constructor** with parameters to initialize the above attributes.

- ☐ Define the following methods:

- start() – prints a message that the car starts.
- Speed() – prints the type of fuel or power used.
- Outlook() – prints the color of the car.
- Price() – prints the model name and price of the car.

- ☐ In the main() method:

- Create a Car object c1 with sample values.
 - Call all the methods to display the car's properties and behavior.
-

3. Pseudocode

CLASS Car

VARIABLES:

model_name

cost

type

color

CONSTRUCTOR Car(a, b, c, d)

SET model_name = a

SET cost = b

SET type = c

SET color = d

METHOD start()

PRINT model_name + " starts with Some sound"

METHOD Speed()

PRINT "My car runs on " + type

METHOD Outlook()

PRINT "My car has " + color + " color"

METHOD Price()

PRINT "My car " + model_name + " costs " + cost

MAIN METHOD

CREATE object c1 of class Car with ("XUV", 30000000, "petrol", "Red")

CALL c1.start()

CALL c1.Speed()

CALL c1.Outlook()

CALL c1.Price()

4. Program Code

```
public class Car{
    String model_name;
    int cost;
    String type;
    String color;

    Car(String a, int b, String c, String d){
        model_name=a;
        cost=b;
        type=c;
        color=d;
    }
    public void start(){
        System.out.println(model_name+" starts with Some sound");
    }
    public void Speed(){
        System.out.println("My car runs on "+type);
    }
    public void Outlook(){
        System.out.println("My car has "+color+ " color");
    }
    public void Price(){
        System.out.println("My car "+model_name+"costs "+cost);
    }
    public static void main(String[] args) {
        Car c1= new Car("XUV",30000000,"petrol","Red");
        c1.start();
        c1.Speed();
        c1.Outlook();
        c1.Price();
    }
}
```

5. Screenshots of Output

```
ata\Roaming\Code\User\workspaceSt  
r'  
XUV starts with Some sound  
My car runs on petrol  
My car has Red color  
My car XUVcosts 30000000  
PS C:\Users\Admin\OneDrive\Desktop
```

6. Observation / Reflection

By creating an object. We are calling the method .previously we are calling method by method name .now we are calling it by creating an object c1

Problem Solving problem 2.1

1. Create two Dog objects:

Dog1: "Buddy", "Golden Retriever", 5

Dog2: "Lucy", "Poodle", 2

2. Algorithm

- ☐ **Define a class** B2_1 with attributes: name, breed, and age.
- ☐ Create a **constructor** that takes three parameters (n, b, a) and assigns them to the instance variables.
- ☐ Define a method bark() that prints the dog's name along with a bark message.
- ☐ Define a method displayInfo() that prints the dog's name and age.

□ In the main() method:

- Create two objects dog1 and dog2 with different values.
- Call bark() and displayInfo() for both dogs

3. Pseudocode

CLASS B2_1

VARIABLES:

name

breed

age

CONSTRUCTOR(n, b, a)

SET name = n

SET breed = b

SET age = a

METHOD bark()

PRINT name + " says: Woof woof!"

METHOD displayInfo()

PRINT "Name: " + name + ", Age: " + age

MAIN METHOD

CREATE dog1 = new B2_1("Blacky", "pomeranian", 7)

CREATE dog2 = new B2_1("milky", "Golden Retriever", 3)

CALL dog1.bark()

CALL dog1.displayInfo()

CALL dog2.bark()

CALL dog2.displayInfo()

4. Program Code

```
public class B2_1 {
    String name;
    String breed;
    int age;

    // Constructor
    B2_1 (String n, String b, int a) {
        name = n;
        breed = b;
        age = a;
    }

    // Method
    void bark() {
        System.out.println(name + " says: Woof woof!");
    }

    // Method to display dog info
    void displayInfo() {
        System.out.println("Name: " + name + ", Age: " + age);
    }

    // Main method
    public static void main(String[] args) {
        B2_1 dog1 = new B2_1 ("Blacky", "pomeranian", 7);
        B2_1 dog2 = new B2_1 ("milky", "Golden Retriever", 3);

        dog1.bark();
        dog1.displayInfo();

        dog2.bark();
        dog2.displayInfo();
    }
}
```

6. Screenshots of Output

```
spot\bin\java.exe' '-XX:
kspaceStorage\6ec1f0d041
Blacky says: woof!
Name: Blacky, Age: 7
milky says: woof!
Name: milky, Age: 3
PS C:\Users\Admin\OnePri
```

7. Observation / Reflection

I learnt how to create an object for the same method twice and know I am able to create object for any class

Problem Solving Problem 2.2

1. Manage Books

Create two Book objects with constructor

Implement displayStatus()

```
void displayStatus() {}
```

```
String status = isopen? "Open": "Closed";
```

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

2. Algorithm

- ☐ Define a class B2_2 to represent a book.
- ☐ Declare instance variables: title, author, isOpen, and status.

- ☐ Create a constructor to initialize title, author, and isOpen.
- ☐ Define a method displayStatus():
 - Use a **ternary operator** to determine if the book is "Open" or "Closed" based on isOpen.
 - Print the title, author, and current status.
- ☐ In the main() method:
 - Create two book objects with different values for isOpen.
 - Call displayStatus() on both to show their states

3. Pseudocode

CLASS B2_2

VARIABLES:

title

author

isOpen

status

CONSTRUCTOR(t, a, o)

SET title = t

SET author = a

SET isOpen = o

METHOD displayStatus()

IF isOpen IS TRUE

SET status = "Open"

ELSE

SET status = "Closed"

PRINT title + " by " + author + " is " + status

MAIN METHOD




```
CREATE book1 = new B2_2("XYZ", "kuvempu", true)
```

```
CREATE book2 = new B2_2("ABC", "James", false)
```

```
CALL book1.displayStatus()
```

```
CALL book2.displayStatus()
```

4. Program Code

```
public class B2_2 {
    String title;
    String author;
    boolean isOpen;
    String status;

    // Constructor
    B2_2(String t, String a, boolean o) {
        title = t;
        author = a;
        isOpen = o;
    }

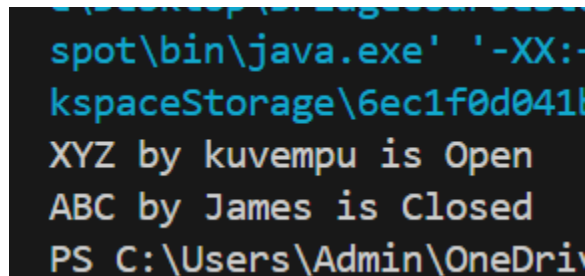
    // Method to display book status
    void displayStatus() {
        /*void displayStatus() {
        // String status = isOpen ? "Open" :
"Closed";
        // System.out.println(title + " by "
+ author + " is " + status);
        // }/

        if (isOpen) {
            status = "Open";
        } else {
            status = "Closed";
        }
        System.out.println(title + " by " + author + " is " + status);
    }

    // Main method
    public static void main(String[] args) {
        B2_2 book1 = new B2_2("XYZ", "kuvempu", true);
        B2_2 book2 = new B2_2("ABC", "James", false);
    }
}
```

```
        book1.displayStatus();  
        book2.displayStatus();  
    }  
}
```

6. Screenshots of Output



```
spot\bin\java.exe' '-XX:-  
kspaceStorage\6ec1f0d041f  
XYZ by kuvempu is Open  
ABC by James is Closed  
PS C:\Users\Admin\OneDrive
```

7. Observation / Reflection

I learnt how to implement the ternary operator to check the condition

Problem Solving Problem 2.3

1. Student Record

Print the records of the student using this keyword and Encapsulation

2. Algorithm

1. Define a class B2_3 with attributes: name, idNumber, and major.
2. Create a constructor B2_3 that initializes name, idNumber, and major.

3. Define a method getInfo() that returns a string containing the student's information.

4. In the main method:

- Create multiple instances of the B2_3 class with different student data.
- Call the getInfo() method for each instance and print the result.

3. Pseudocode

CLASS B2_3

ATTRIBUTES

name

idNumber

major

CONSTRUCTOR B2_3(name, idNumber, major)

INITIALIZE name, idNumber, major

METHOD getInfo()

RETURN name + ", ID: " + idNumber + ", Major: " + major

MAIN

CREATE B2_3 s1 WITH name = "sameer", idNumber = "c1", major = "Cse"

CREATE B2_3 s2 WITH name = "Sahana", idNumber = "m1", major = "Mechanical Engineering"

CREATE B2_3 s3 WITH name = "rahan", idNumber = "e1", major = "Electronics"

PRINT s1.getInfo()

PRINT s2.getInfo()

PRINT s3.getInfo()

4. Program Code

```
public class B2_3 {  
  
    String name;
```

```
String idNumber;
String major;

public B2_3(String name, String idNumber, String major) {
    this.name = name;
    this.idNumber = idNumber;
    this.major = major;
}

public String getInfo() {
    return name + ", ID: " + idNumber + ", Major: " + major;
}

public static void main(String[] args) {
    B2_3 s1 = new B2_3("sameer", "c1", "Cse");
    B2_3 s2 = new B2_3("Sahana", "m1", "Mechanical Engineering");
    B2_3 s3 = new B2_3("rahan", "e1", "Electronics");
    System.out.println(s1.getInfo());
    System.out.println(s2.getInfo());
    System.out.println(s3.getInfo());
}
}
```

5. Test Cases

Test Case No.	Input	Expected Output	Actual Output	Status (Pass/Fail)
1	Sameer	Sameer	Sameer	pass
2	Sahana	Sahana	Sahana	pass
3	rahan	Rahan	Rahan	pass

6. Screenshots of Output

```
sameer, ID: c1, Major: Cse  
Sahana, ID: m1, Major: Mechanical Engineering  
rahan, ID: e1, Major: Electronics  
PS C:\Users\Admin\OneDrive\Desktop\BridgecourseStemu
```

7. Observation / Reflection

Encapsulation: The code effectively encapsulates student attributes (name, idNumber, and major) within the class.

Problem Solving Problem3.1

1. Bank Account I

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

Try invalid operations (e.g., negative deposit, excessive withdrawal)

2. Algorithm

1. Define a class C3_1 with an instance attribute balance.
2. Create a constructor C3_1 that:
 - Initializes the balance with a valid initial amount.
 - Sets the balance to 0 if the initial amount is negative.
3. Define methods:
 - getBalance(): Returns the current balance.
 - deposit(amount): Adds a valid amount to the balance and prints a confirmation message.

- withdraw(amount): Subtracts a valid amount from the balance if sufficient funds are available and prints a confirmation message.

4. In the main method:

- Create an instance of the C3_1 class with an initial balance.
- Perform deposit and withdrawal operations with different amounts.
- Print the initial and final balances.

3. Pseudocode

CLASS C3_1

ATTRIBUTE

balance

CONSTRUCTOR C3_1(initialBalance)

IF initialBalance \geq 0 THEN

SET balance TO initialBalance

ELSE

PRINT "Initial balance can't be negative. Setting to 0."

SET balance TO 0

METHOD getBalance()

RETURN balance

METHOD deposit(amount)

IF amount > 0 THEN

ADD amount TO balance

PRINT "Deposited: " + amount

ELSE

PRINT "Invalid deposit amount."

METHOD withdraw(amount)

```
IF amount <= 0 THEN
    PRINT "Invalid withdrawal amount."
ELSE IF amount > balance THEN
    PRINT "Insufficient funds. Withdrawal denied."
ELSE
    SUBTRACT amount FROM balance
    PRINT "Withdrawn: " + amount

MAIN
    CREATE C3_1 account WITH initialBalance = 1000.0
    PRINT "Initial Balance: " + account.getBalance()
    CALL account.deposit(500.0)
    CALL account.deposit(-100.0)
    CALL account.withdraw(200.0)
    CALL account.withdraw(2000.0)
    CALL account.withdraw(-50.0)
    PRINT "Final Balance: " + account.getBalance()
```

4. Program Code

```
public class C3_1 {
    // Instance attribute
    private double balance;

    // Constructor
    public C3_1(double initialBalance) {
        if (initialBalance >= 0) {
            this.balance = initialBalance;
        } else {
            System.out.println("Initial balance can't be negative. Setting to 0.");
            this.balance = 0;
        }
    }
}
```

```
public double getBalance() {
    return balance;
}
public void deposit(double amount) {
    if (amount > 0) {
        balance += amount;
        System.out.println("Deposited: " + amount);
    } else {
        System.out.println("Invalid deposit amount.");
    }
}
public void withdraw(double amount) {
    if (amount <= 0) {
        System.out.println("Invalid withdrawal amount.");
    } else if (amount > balance) {
        System.out.println("Insufficient funds. Withdrawal denied.");
    } else {
        balance -= amount;
        System.out.println("Withdrawn: " + amount);
    }
}
public static void main(String[] args) {
    C3_1 account = new C3_1(1000.0);

    System.out.println("Initial Balance: " + account.getBalance());

    account.deposit(500.0);
    account.deposit(-100.0);

    account.withdraw(200.0);
    account.withdraw(2000.0);
    account.withdraw(-50.0);

    System.out.println("Final Balance: " + account.getBalance());
}
```

5. Screenshots of Output

```
Invalid deposit amount.  
Withdrawn: 200.0  
Insufficient funds. Withdrawal denied.  
Invalid withdrawal amount.  
Final Balance: 1300.0  
PS C:\Users\Admin\OneDrive\Desktop\Bridge
```

6. Observation / Reflection

- 1. Encapsulation:** The code effectively encapsulates the balance attribute within the C3_1 class.
- 2. Data Validation:** The constructor and methods (deposit and withdraw) validate input data to prevent invalid operations.
- 3. Error Handling:** The code handles errors like negative initial balance, invalid deposit/withdrawal amounts, and insufficient funds.

Problem Solving Problem 3.2

1. Product Inventory

Problem 3.2: Product Inventory

```
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 void setQuantity(int quantity) {

if (quantity >= 0) {

this.quantity = quantity;

}

}

public double getTotalValue() {

return price*quantity;

}
```

Test object creation, use getters/setters, validate constraints

2. Algorithm

1. Define a class C3_2 with private instance variables name, price, and quantity.
2. Create a constructor C3_2 that initializes the product with a name, price, and quantity.
3. Define getter methods for name, price, and quantity.
4. Define setter methods for price and quantity with validation checks.

5. Define a method getTotalValue() to calculate the total value of the product.

6. In the main method:

- Create a product instance with initial details.
 - Print the initial details and total value.
 - Attempt to set invalid price and quantity values.
 - Update the product with valid values and print the updated details.
-

3. Pseudocode

CLASS C3_2

ATTRIBUTES

name

price

quantity

CONSTRUCTOR C3_2(name, price, quantity)

INITIALIZE name, price, quantity

CALL setPrice(price)

CALL setQuantity(quantity)

METHOD getName()

RETURN name

METHOD getPrice()

RETURN price

METHOD getQuantity()

RETURN quantity

METHOD setPrice(price)

IF price > 0 THEN

SET price TO price

ELSE



PRINT "Invalid price. Must be positive."

METHOD setQuantity(quantity)

IF quantity >= 0 THEN

SET quantity TO quantity

ELSE

PRINT "Invalid quantity. Cannot be negative."

METHOD getTotalValue()

RETURN price * quantity

MAIN

CREATE C3_2 product WITH name = "Laptop", price = 45000.0, quantity = 5

PRINT product.getName(), product.getPrice(), product.getQuantity(), product.getTotalValue()

CALL product.setPrice(-1000)

CALL product.setQuantity(-2)

CALL product.setPrice(50000)

CALL product.setQuantity(3)

PRINT product.getPrice(), product.getQuantity(), product.getTotalValue()

4. Program Code

```
public class C3_2 {  
    // Private instance variables  
    private String name;  
    private double price;  
    private int quantity;  
  
    // Constructor  
    public C3_2(String name, double price, int quantity) {  
        this.name = name;  
        setPrice(price);  
        setQuantity(quantity);  
    }  
}
```

```
}

// Getter for name
public String getName() {
    return name;
}

// Getter for price
public double getPrice() {
    return price;
}

// Getter for quantity
public int getQuantity() {
    return quantity;
}

// Setter for price with validation
public void setPrice(double price) {
    if (price > 0) {
        this.price = price;
    } else {
        System.out.println("Invalid price. Must be positive.");
    }
}

// Setter for quantity with validation
public void setQuantity(int quantity) {
    if (quantity >= 0) {
        this.quantity = quantity;
    } else {
        System.out.println("Invalid quantity. Cannot be negative.");
    }
}

// Method to calculate total value
public double getTotalValue() {
    return price * quantity;
}

// Main method for testing
public static void main(String[] args) {
    // Create a product
    C3_2 product = new C3_2("Laptop", 45000.0, 5);
```

```
// Print initial details
System.out.println("Product: " + product.getName());
System.out.println("Price: " + product.getPrice());
System.out.println("Quantity: " + product.getQuantity());
System.out.println("Total Value: " + product.getTotalValue());

// Try setting invalid values
product.setPrice(-1000);    // Invalid
product.setQuantity(-2);    // Invalid

// Update with valid values
product.setPrice(50000);
product.setQuantity(3);

// Print updated details
System.out.println("\nUpdated Details:");
System.out.println("Price: " + product.getPrice());
System.out.println("Quantity: " + product.getQuantity());
System.out.println("Total Value: " + product.getTotalValue());
}
```



5. Screenshots of Output

```
Updated Details:
Price: 50000.0
Quantity: 3
Total Value: 150000.0
PS C:\Users\Admin\OneDrive\Desktop\Bridg
```

6. Observation / Reflection

Class Name: Consider renaming the class to something more descriptive, such as Product.

Additional Features: Add features like product categorization, inventory management, or support for multiple product types.

Error Handling: Consider throwing exceptions instead of printing error messages to handle invalid data more robustly.

