ASSIGNMENT DAY-5

Name: Vishnu Priya

Enrolment Number: SU625MR009

Batch/Class: MERN stack

Assignment: Day 5

Date of Submission:30th June 2025

Problem Statement:

Simple Dog: Create a class with name simple dog and create 2 methods with name Dog and bark within it. Print the species and numbers of legs.

Algorithm:

Step 1: Start

Step 2: Create class with name SimpleDog

Step 3: Include attributes, constructor and bark

Step 4: Create an object for SimpleDog in main class

Step 5: Print the species and number of legs

Step 6: End

Pseudo Code:

```
Class SimpleDog:
    Define Static Nested Class Dog:
        Attributes:
            name        → String
            breed       → String
            age         → Integer
            species     → Static String = "Canis Familiaris"
            numlegs     → Static Integer = 4
        Constructor(name, breed, age):
            Set this.name = name
            Set this.breed = breed
```

Set this.age = age

Method bark():

Print "Woof!"

Method main():

Create an object myDog of class Dog with name="Buddy", breed="Golden Retriever", age=3

Print "species: " + Dog.species

Print "legs: " + Dog.numlegs

Call myDog.bark()

Code:

```
package Classes;

public class SimpleDog {
    public class Dog{
        String name;
        String breed;
        int age;
        static String species="Canis Familiaris";
        static int numlegs=4;
        public Dog(String name,String breed,int age) {
                this.name=name;
                this.breed=breed;
                this.age=age;
        }
        void bark() {
                System.out.println("Woof!");
        }
   }
        public static void main(String[] args) {
                // TODO Auto-generated method stub
                SimpleDog outer = new SimpleDog();
                Dog myDog=outer.new Dog("Buddy","golden retriever",3);
      System.out.println("species: "+Dog.species);
      System.out.println("legs: "+Dog.numlegs);
      myDog.bark();
        }

}
```

Test Cases:

Test Case no.	Input	Expected Output	Actual Output	Status(Pass/Fail)

TC1	Buddy Golden Retriever 3	species: Canis Familiaris legs: 4 Woof! Buddy golden retriever 3	species: Canis Familiaris legs: 4 Woof! Buddy golden retriever 3	Pass
TC2	Toby German Shepherd 8	species: Canis Familiaris legs: 4 Woof! Toby German Shepherd 8	species: Canis Familiaris legs: 4 Woof! Toby German Shepherd 8	Pass
TC3	Leela Jay Hari	Hello, leela! Hello, jay! Hello, hari!	Hello, leela! Hello, jay! Hello, hari!	Pass

Output:

TC1:

```
species: Canis Familiaris
legs: 4
Woof!
Buddy
golden retriever
8
```

TC2:

```
species: Canis Familiaris
legs: 4
Woof!
Toby
German Shepherd
8
```

TC3:

```
species: Canis Familiaris
legs: 4
Woof!
Banty
Pug
5
```

Observation:

- Understanding the class and object working and creating constructor

Problem Statement:

Basic Book: Write a program to create a class book and add attributes of title, author, num_pages and methods to OpenBook(), closeBook()

Algorithm:

Step 1: Start

Step 2: Create a function that takes integer as an input

Step 3: Return the square of the number and close the function

Step 4: Seek the input from the user using scanner class

Step 5: Pass the number n to the function and call the function.

Step 6: End

Pseudo Code:

```
CLASS BasicBook:
    CLASS Book:
        DECLARE title AS String
        DECLARE author AS String
        DECLARE numPages AS Integer
        DECLARE isOpen AS Boolean
        METHOD Constructor(title, author, numPages):
            SET this.title = title
            SET this.author = author
            SET this.numPages = numPages
            SET this.isOpen = false
        METHOD openBook():
            SET isOpen = true
            PRINT "Book is now open."
        METHOD closeBook():
            SET isOpen = false
            PRINT "Book is now closed."
    METHOD main():
        CREATE bb AS OBJECT OF BasicBook
        CREATE myBook AS OBJECT OF Book USING bb WITH ("Java Basics", "Alice", 250)
        PRINT myBook.title
        PRINT myBook.author
        PRINT myBook.numPages
        CALL myBook.openBook()
        CALL myBook.closeBook()
```

Code:

```
package Classes;
public class BasicBook {
```

```
public class Book {
    String title;
    String author;
    int numPages;
    boolean isOpen;
    public Book(String title, String author, int numPages) {
        this.title = title;
        this.author = author;
        this.numPages = numPages;
        this.isOpen = false;
    }
    void openBook() {
        isOpen = true;
        System.out.println("Book is now open.");
    }
    void closeBook() {
        isOpen = false;
        System.out.println("Book is now closed.");
    }
}
    public static void main(String[] args) {
            BasicBook bb=new BasicBook();
        Book myBook = bb.new Book("Java Basics", "Alice", 250);
        System.out.println(myBook.title);
        System.out.println(myBook.author);
        System.out.println(myBook.numPages);
        myBook.openBook();
        myBook.closeBook();
    }
}
```

Output:

```
Java Basics
Alice
250
Book is now open.
Book is now closed.
```

Observation:

- In the program, the working of return statement and using the print statement and function calling in same line.

Problem Statement:

Dogs: Create a class named as dog with method bark() and print details of 2 dogs using object

Algorithm:

Step 1: Start

Step 2: Create class with name SimpleDog

Step 3: Include attributes, constructor bark() and printDetails()

Step 4: Create an object for Dog in main class

Step 5: Call the constructor and print the output

Step 6: End

Pseudo Code:

```
CLASS Dogs:
    STATIC CLASS Dog:
        DECLARE name AS String
        DECLARE breed AS String
        DECLARE age AS Integer
        METHOD Constructor(name, breed, age):
            SET this.name = name
            SET this.breed = breed
            SET this.age = age
        METHOD bark():
            PRINT name + " says: Woof!"
        METHOD printDetails():
            PRINT "Name: " + name + ", Breed: " + breed + ", Age: " + age
    METHOD main():
        CREATE dog1 AS Dog WITH ("Buddy", "Golden Retriever", 5)
        CREATE dog2 AS Dog WITH ("Lucy", "Poodle", 2)
        CALL dog1.bark()
        CALL dog1.printDetails()
        CALL dog2.bark()
        CALL dog2.printDetails()
```

Code:

```java
package Objects;

public class Dogs {
        static class Dog {
    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(name + " says: Woof!");
    }

    void printDetails() {
        System.out.println("Name: " + name + ", Breed: " + breed + ", Age: " + age);
    }
}

public static void main(String[] args) {
    Dog dog1 = new Dog("Buddy", "Golden Retriever", 5);
    Dog dog2 = new Dog("Lucy", "Poodle", 2);

    dog1.bark();
    dog1.printDetails();

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

Output:

```
Buddy says: Woof!
Name: Buddy, Breed: Golden Retriever, Age: 5
Lucy says: Woof!
Name: Lucy, Breed: Poodle, Age: 2
```

Problem Statement:

Manage Books: Create 2 book objective with constructor that display the details of the book as open or close.

Algorithm:

Step 1: Start

Step 2: Create class with name ManageBook

Step 3: Include attributes, constructor and isOpen

Step 4: Create an object for ManageBook in main class

Step 5: Print the book details and status

Step 6: End

Pseudo Code:

```
CLASS ManageBook:
    STATIC CLASS Book:
        DECLARE title AS String
        DECLARE author AS String
        DECLARE numPages AS Integer
        DECLARE isOpen AS Boolean
        METHOD Constructor(title, author, numPages, isOpen):
            SET this.title = title
            SET this.author = author
            SET this.numPages = numPages
            SET this.isOpen = isOpen
        METHOD displayStatus():
            IF isOpen IS TRUE:
                SET status = "Open"
            ELSE:
                SET status = "Closed"
            PRINT title + " by " + author + " is " + status
    METHOD main():
        CREATE book1 AS Book WITH ("Java Programming", "Alice Smith", 300, true)
        CREATE book2 AS Book WITH ("Data Structures", "Bob Jones", 250, false)
        CALL book1.displayStatus()
        CALL book2.displayStatus()
```

Code:

```
package Objects;
```

```
public class ManageBook {
        static class Book {
            String title;
            String author;
            int numPages;
            boolean isOpen;

            Book(String title, String author, int numPages, boolean isOpen) {
                this.title = title;
                this.author = author;
                this.numPages = numPages;
                this.isOpen = isOpen;
            }

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

        public static void main(String[] args) {
            Book book1 = new Book("Java Programming", "Alice Smith", 300, true);
            Book book2 = new Book("Data Structures", "Bob Jones", 250, false);

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

Output:

```
Java Programming by Alice Smith is Open
Data Structures by Bob Jones is Closed
```

Problem Statement:

Student Record: Create three Student objects and print their info.

Algorithm:

Step 1: Start

Step 2: Create a Class student with 3 attributes (name,idnumber, and major)

Step 3: create a constructor to initialise these attributes with given values

Step 4: Create a method getInfo() that returns a formatted string combining name,ID and major

Step 5: In the main method, create three student objects with appropriate values

Step 6: call the method and print the information

Step 7: End

Pseudo Code:

```
CLASS StudentRecord:
    STATIC CLASS Student:
        DECLARE name AS String
        DECLARE idNumber AS String
        DECLARE major AS String
        METHOD Constructor(name, idNumber, major):
            SET this.name = name
            SET this.idNumber = idNumber
            SET this.major = major
        METHOD getInfo():
            RETURN name + ", ID: " + idNumber + ", Major: " + major
    METHOD main():
        CREATE s1 AS Student WITH ("Aarna", "S001", "Computer Science")
        CREATE s2 AS Student WITH ("Rahul", "S002", "Mechanical")
        CREATE s3 AS Student WITH ("Sneha", "S003", "Electronics")
        PRINT s1.getInfo()
        PRINT s2.getInfo()
        PRINT s3.getInfo()
```

Code:

```java
package Objects;
public class StudentRecord{
static 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 getInfo() {
        return name + ", ID: " + idNumber + ", Major: " + major;
    }
}

public static void main(String[] args) {
    Student s1 = new Student("Aarna", "S001", "Computer Science");
    Student s2 = new Student("Rahul", "S002", "Mechanical");
    Student s3 = new Student("Sneha", "S003", "Electronics");

    System.out.println(s1.getInfo());
    System.out.println(s2.getInfo());
    System.out.println(s3.getInfo());
}

}
```

Output:

```
Aarna, ID: S001, Major: Computer Science
Rahul, ID: S002, Major: Mechanical
Sneha, ID: S003, Major: Electronics
```

Problem Statement:

Bank Account: Create bankAccount class with methods like getBalance(), deposit(), and withdraw() methods. Try to execute with invalid operation.

Algorithm:

Step 1: Start
Step 2: Create a class BankAccount with a private variable balance
Step 3: Create a constructor that takes initialBalance
Step 4: If initialBalance is non-negative, set balance to it; otherwise, set balance to 0 and print a warning
Step 5: Create a method getBalance() that returns the current balance
Step 6: Create a method deposit(amount)
- If amount > 0, add it to balance and print deposit message
- Else, print invalid amount message

Step 7: Create a method withdraw(amount)
- If amount > 0 and amount <= balance, subtract it and print withdrawal message
- Else, print invalid or excessive withdrawal message

Step 8: In main(), create a BankAccount object with initial balance = 1000
Step 9: Print current balance using getBalance()
Step 10: Call deposit(500) and deposit(-100)
Step 11: Call withdraw(300), withdraw(1500), and withdraw(-200)
Step 12: Print final balance using getBalance()
Step 13: End

Pseudo Code:

```
CLASS BankAccount:

    PRIVATE VARIABLE balance AS Double

    METHOD Constructor(initialBalance):

        IF initialBalance >= 0 THEN

            SET balance = initialBalance

        ELSE

            PRINT "Invalid initial balance. Setting to 0."

            SET balance = 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 AND amount <= balance THEN
            SUBTRACT amount FROM balance
            PRINT "Withdrawn: " + amount
        ELSE
            PRINT "Invalid or excessive withdrawal amount."
METHOD main():
    CREATE account AS BankAccount WITH initialBalance = 1000
    PRINT "Current Balance: " + CALL account.getBalance()
   CALL account.deposit(500)
    CALL account.deposit(-100)
    CALL account.withdraw(300)
    CALL account.withdraw(1500)
    CALL account.withdraw(-200)
    PRINT "Final Balance: " + CALL account.getBalance()
```

Code:

```java
package Encapsulation;
public class BankAccount {
            private double balance;
            public BankAccount(double initialBalance) {
               if (initialBalance >= 0) {
                  balance = initialBalance;
               } else {
                  System.out.println("Invalid initial balance. Setting to 0.");
                  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 && amount <= balance) {
            balance -= amount;
            System.out.println("Withdrawn: " + amount);
        } else {
            System.out.println("Invalid or excessive withdrawal amount.");
        }
    }
    public static void main(String[] args) {
        BankAccount account = new BankAccount(1000);

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

        account.deposit(500);
        account.deposit(-100);

        account.withdraw(300);
        account.withdraw(1500);
        account.withdraw(-200);

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

Output:

```
Deposited: 500.0
Invalid deposit amount.
Withdrawn: 300.0
Invalid or excessive withdrawal amount.
Invalid or excessive withdrawal amount.
Final Balance: 1200.0
```

Problem Statement:

Product Inventory: For the give code, test object creation, use getters/setters, validate constraints

Algorithm:

Step 1: Start

Step 2: Create a class ProductInventory with private variables name, price, and quantity

Step 3: Create a constructor to initialize values and call setters to apply validation

Step 4: Create getter methods: getName(), getPrice(), and getQuantity()

Step 5: Create setPrice() method

- If price > 0, set the value
- Else, print invalid price message

Step 6: Create setQuantity() method

- If quantity >= 0, set the value
- Else, print invalid quantity message

Step 7: Create getTotalValue() method that returns price * quantity

Step 8: In main() method:

- Create a ProductInventory object with name "Laptop", price 50000, quantity 2
- Print product details and total value
- Call setPrice() with -1000
- Call setQuantity() with -5
- Print updated price and quantity

Step 9: End

Pseudo Code:

```
CLASS ProductInventory:
    PRIVATE name AS String
    PRIVATE price AS Double
    PRIVATE quantity AS Integer
    METHOD Constructor(name, price, quantity):
        SET this.name = name
```

```
        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 this.price = price
        ELSE
            PRINT "Invalid price. Must be > 0."
    METHOD setQuantity(quantity):
        IF quantity >= 0 THEN
            SET this.quantity = quantity
        ELSE
            PRINT "Invalid quantity. Must be >= 0."
    METHOD getTotalValue():
        RETURN price * quantity
METHOD main():
    CREATE p1 AS ProductInventory WITH ("Laptop", 50000, 2)
    PRINT name, price, quantity, and total value
    CALL p1.setPrice(-1000)
    CALL p1.setQuantity(-5)
    PRINT updated price and quantity
```

Code:

```java
package Encapsulation;
public class ProductInventory {
    private String name;
    private double price;
    private int quantity;

    public ProductInventory(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("Invalid price. Must be > 0.");
        }
    }

    public void setQuantity(int quantity) {
        if (quantity >= 0) {
            this.quantity = quantity;
        } else {
            System.out.println("Invalid quantity. Must be >= 0.");
        }
    }

    public double getTotalValue() {
        return price * quantity;
    }

    public static void main(String[] args) {
            ProductInventory p1 = new ProductInventory("laptop", 50000.0, 2);
        System.out.println("Product: " + p1.getName());
        System.out.println("Price: " + p1.getPrice());
        System.out.println("Quantity: " + p1.getQuantity());
        System.out.println("Total Value: " + p1.getTotalValue());

        p1.setPrice(-1000);
        p1.setQuantity(-5);
```

```
        System.out.println("Updated Price: " + p1.getPrice());
        System.out.println("Updated Quantity: " + p1.getQuantity());
        }
}
```

Output:

TC1:

```
Product: laptop
Price: 50000.0
Quantity: 2
Total Value: 100000.0
Invalid price. Must be > 0.
Invalid quantity. Must be >= 0.
Updated Price: 50000.0
Updated Quantity: 2
```