**Learner Assignment Submission Format**

**Learner Details**

* **Name: Nithya B A**
* **Enrollment Number: SU625MR010**
* **Batch / Class: Mern stack**
* **Assignment: (Bridge Course Day 5)**
* **Date of Submission: 30/6/2025**

**Problem Solving Activity 1.1**

1. **Program Statement**

**Write a program to create a class of dog and object of bark and also print number of legs, species**

1. **Algorithm**

Step 1: Start

Step 2: Create class and object

Step 3: Introduce attributes and constructors

Step 4: Print the output

Step 5: End

1. **Pseudocode**

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()

1. **Program Code**

**package** Day5;

**import** java.util.Scanner;

**public** **class** SimpleD {

**public** **class** Dog{

String Name;

String breed;

**int** age;

**static** String *species* = "Canis familiaras";

**static** **int** *legs*=4;

**public** Dog(String Name,String breed,**int** age) {

**this**.Name=Name;

**this**.breed=breed;

**this**.age=age;

}

**void** bark() {

System.***out***.println("wolf");

}

}

**public** **static** **void** main(String[]args) {

SimpleD outer=**new** SimpleD();

Dog mydog=outer.**new** Dog("Buddey","golden retriver",3);

System.***out***.println("species:"+Dog.*species*);

System.***out***.println("legs:"+Dog.*legs*);

mydog.bark();

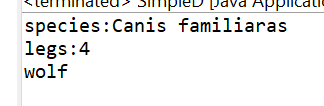
}

}

**5. Test Cases**

|  |  |  |  |  |
| --- | --- | --- | --- | --- |
| Test Case No. | Input | Expected Output | Actual Output | Status (Pass/Fail) |
| 1 | Canis Familiaris  4 | species:Canis familiaras  legs:4  wolf | species:Canis familiaras  legs:4  wolf | Pass |

**6.Screenshots of Output**

****

**7. Observation / Reflection**

Usage of class and object to call the function and to print the output

**Problem Solving Activity 1.2**

1. **Program Statement**

**Write a program to create a class book and add attributes of title, author, num\_pages and method to openBook( ),closeBook( )**

1. **Algorithm**

Step1: Start

Step2: Define BasicBook class with attributes

Step3: Create a construction and method within the class

Step4: Instantiate main class

Step5: Call function and print the output

Step6: End

1. **Pseudocode**

BEGIN

DEFINE class Book

DECLARE title, author, numPages, isOpen

METHOD constructor(title, author, numPages)

SET this.title = title

SET this.author = author

SET this.numPages = numPages

SET this.isOpen = FALSE

END constructor

METHOD openBook()

SET this.isOpen = TRUE

PRINT "Book is now open."

END openBook

METHOD closeBook()

SET this.isOpen = FALSE

PRINT "Book is now closed."

END closeBook

END class Book

DECLARE myBook AS Book

CALL myBook.constructor("Java Basics", "Alice", 250)

CALL myBook.openBook()

CALL myBook.closeBook()

END

1. **Program 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);

myBook.openBook();

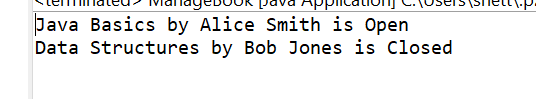
myBook.closeBook();

}

}

}

**5.Screenshots of Output**

****

**7. Observation / Reflection**

**Creating a constructor was a challenge and resolved with the help of other peers**

**Problem Solving Activity 2.1**

1. **Program Statement**

**Write a program to create two Dog objects and a function**

1. **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

1. **Pseudocode**

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()

1. **Program Code**

**package** Day5;

**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();

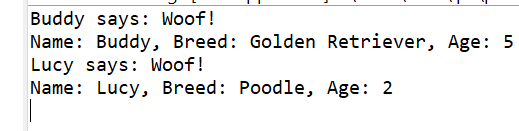
dog2.bark();

dog2.printDetails();

}

}

**5.Screenshots of Output**



**Problem Solving Activity 2.3**

1. **Program Statement**

**Write a JAVA program to create three student objects and print the code**

1. **Algorithm**

Step 1: Start

Step 2: Create a class Student with three attributes: name, idNumber, and major

Step 3: Create a constructor to initialize 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 getInfo() method for each student object

Step 7: Print the returned student information

Step 8: End

1. **Pseudocode**

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()

1. **Program Code**

package Day5;

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 getInfo() {

return name + ", ID: " + idNumber + ", Major: " + major;

}

public static void main(String[] args) {

Student stu1 = new Student("Alice Johnson", "S1001", "Computer Science");

Student stu2 = new Student("Bob Smith", "S1002", "Mathematics");

Student stu3 = new Student("Carol Lee", "S1003", "Physics");

System.*out*.println(stu1.getInfo());

System.*out*.println(stu2.getInfo());

System.*out*.println(stu3.getInfo());

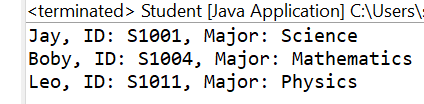
}

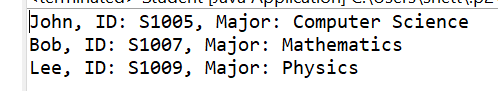
}

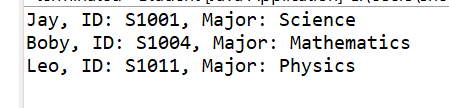
**5. Test Cases**

|  |  |  |  |  |
| --- | --- | --- | --- | --- |
| Test Case No. | Input | Expected Output | Actual Output | Status (Pass/Fail) |
| 1 | Alice S1001  Computer science  Bob Smith  S1002  Mathematics  Carol Lee  S1003  Physics | Alice Johnson, ID: S1001, Major: Computer Science  Bob Smith, ID: S1002, Major: Mathematics  Carol Lee, ID: S1003, Major: Physics | Alice Johnson, ID: S1001, Major: Computer Science  Bob Smith, ID: S1002, Major: Mathematics  Carol Lee, ID: S1003, Major: Physics | Pass |
| 2 | John S1005  Computer Science  Bob S1007  Mathematics  Lee S1009  Physics | John, ID: S1005, Major: Computer Science  Bob , ID: S1007, Major: Mathematics  Lee, ID: S1009, Major: Physics | John, ID: S1005, Major: Computer Science  Bob , ID: S1007, Major: Mathematics  Lee, ID: S1009, Major: Physics | Pass |
| 3 | Jay S1001  Science  Boby S1004  Mathematics  Leo S1011  Physics | Jay, ID: S1001, Major: Science  Boby, ID: S1004, Major: Mathematics  Leo, ID: S1011, Major: Physics | Jay, ID: S1001, Major: Science  Boby, ID: S1004, Major: Mathematics  Leo, ID: S1011, Major: Physics | Pass |

**6.Screenshots of Output**

****

****

****

**7. Observation / Reflection**

Used function call and displayed three students information

**Problem Solving Activity 3.1**

1. **Program Statement**

**Write a program to calculate Bank details with tax**

1. **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

1. **Pseudocode**

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()

1. **Program Code**

**package** Day5;

**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);

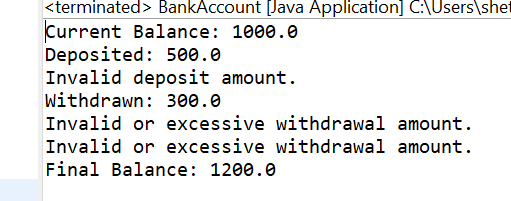
account.withdraw(-200);

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

}

}

**5.Screenshts of Output**

****

**Problem Solving Activity 3.2**

1. **Program Statement**

**Write a program to calculate**

1. **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

1. **Pseudocode**

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

1. **Program Code**

**package** Day5;

**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;

} **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) {

Product p1 = **new** Product("Laptop", 50000, 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);

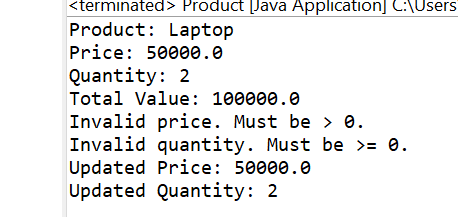
System.***out***.println("Updated Price: " + p1.getPrice());

System.***out***.println("Updated Quantity: " + p1.getQuantity());

}

}

**5.Screenshots of Output**

****

**7. Observation / Reflection**

Created function call for all the different tasks