LAB RECORD

23CSE111 – Object Oriented Programming

Submitted by

CH.SC.U4CSE24106 – **C S Deeraj**

BACHELOR OF TECHNOLOGY

IN

COMPUTER SCIENCE AND ENGINEERING

AMRITA VISHWA VIDYAPEETHAM

AMRITA SCHOOL OF COMPUTING

CHENNAI

March - 2025

AMRITA VISHWA VIDYAPEETHAM

AMRITA SCHOOL OF COMPUTING, CHENNAI

BONAFIDE CERTIFICATE

This is to certify that the Lab Record work for 23CSE111- Object Oriented Programming Subject submitted by ***CH.SC.U4CSE24106 – C S Deeraj*** in **"Computer Science and Engineering"** is a bonafide record of the work carried out under my guidance and supervision at Amrita School of Computing, Chennai.

This Lab examination held on 08/04/2025

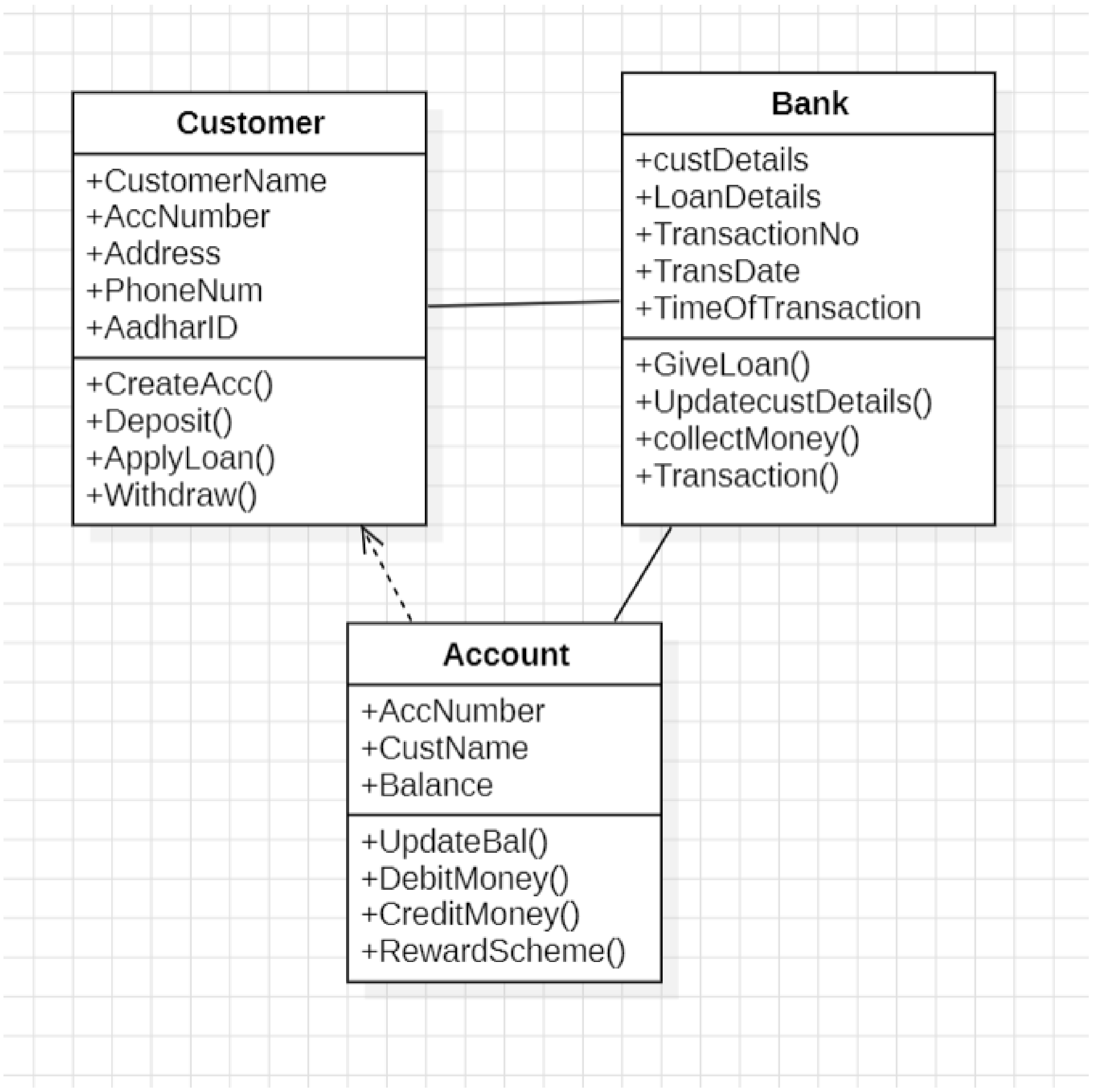
Internal Examiner 1	Internal Examiner 2

Index

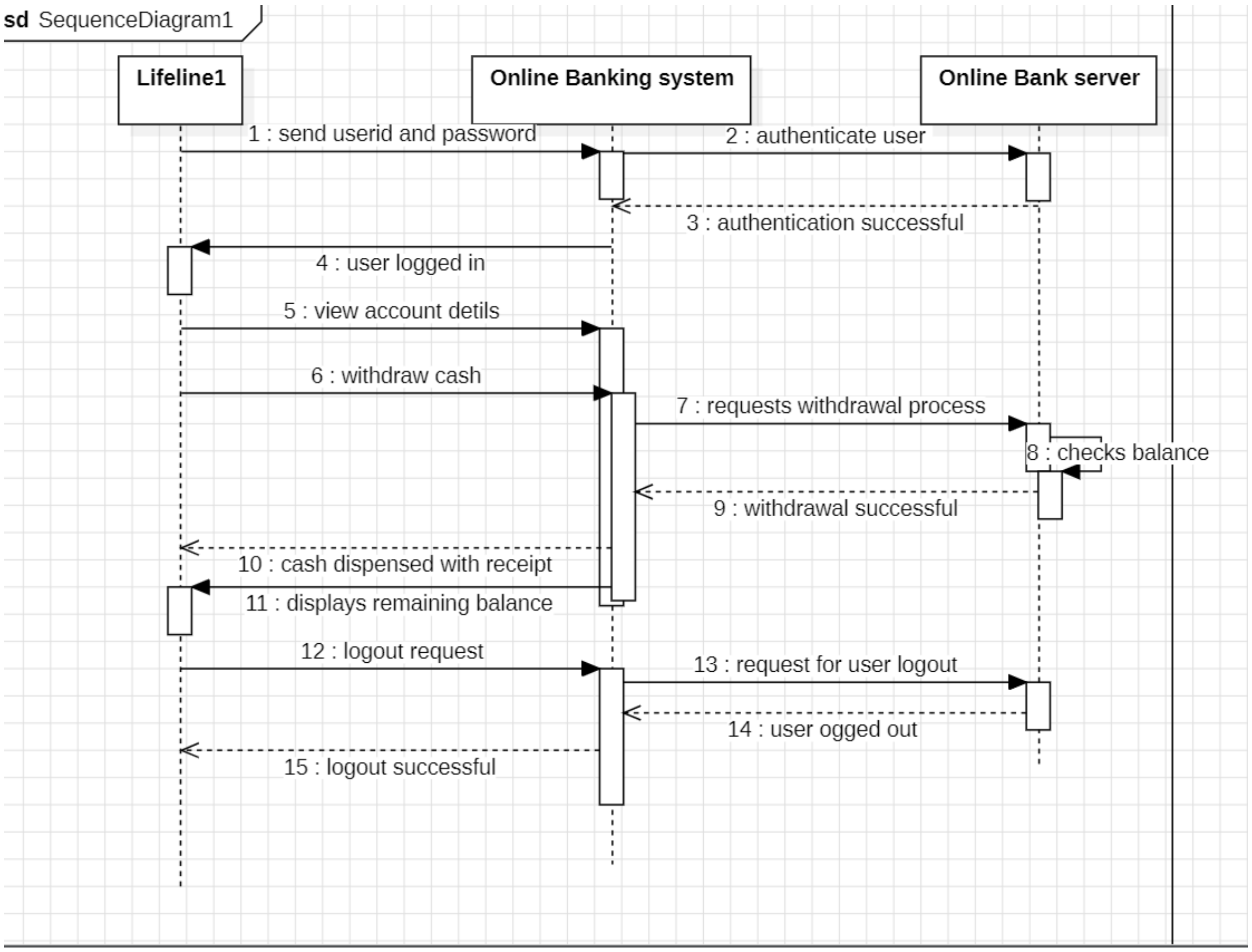
	17.a) Note manager and storing system	
	17.b) Task storing & Management with Timestamp	
	17.c) User login history tracker	
	17.d) Word Search in a written Document	

1. Bank Management System UML:

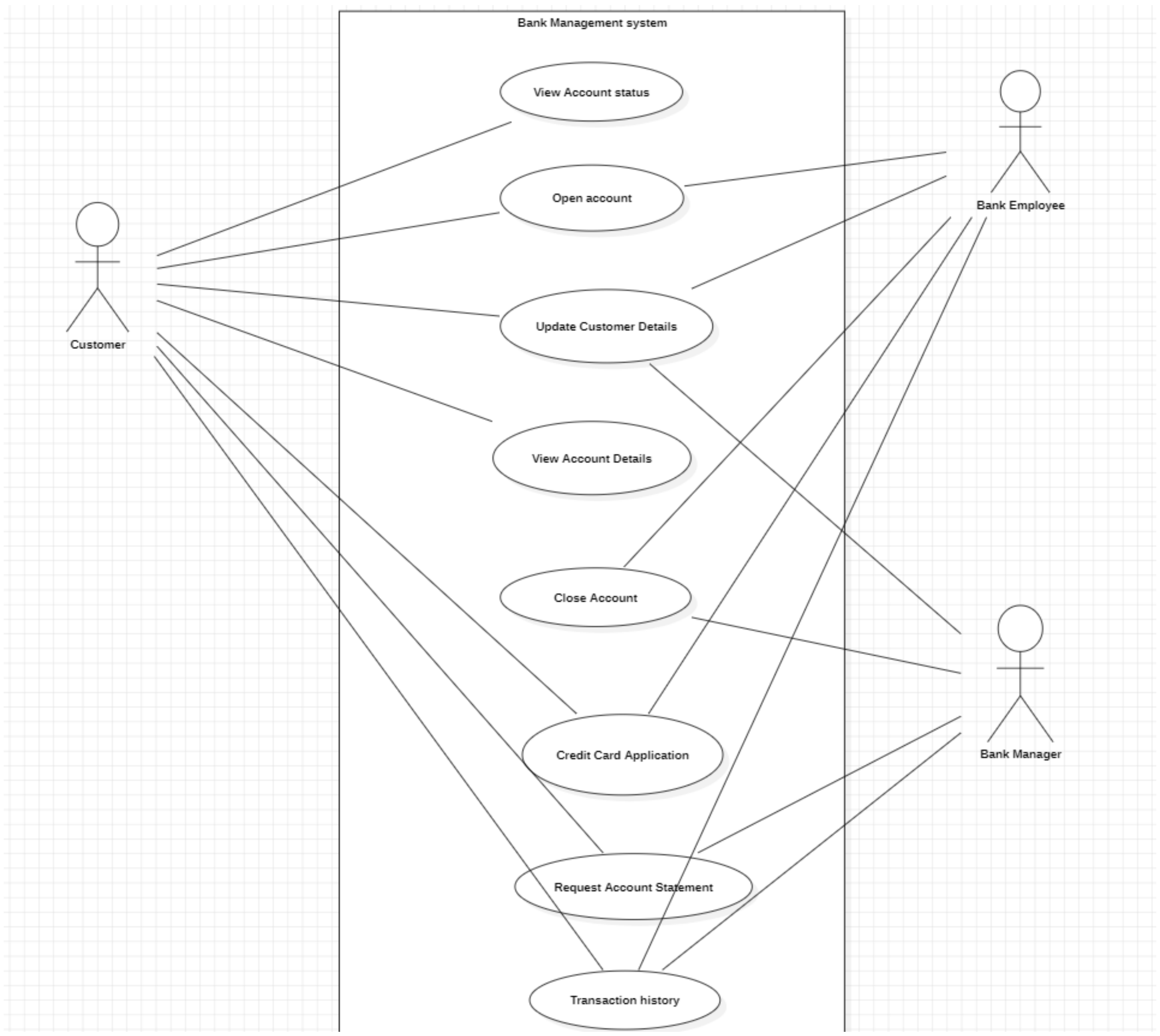
(a) Class Diagram:



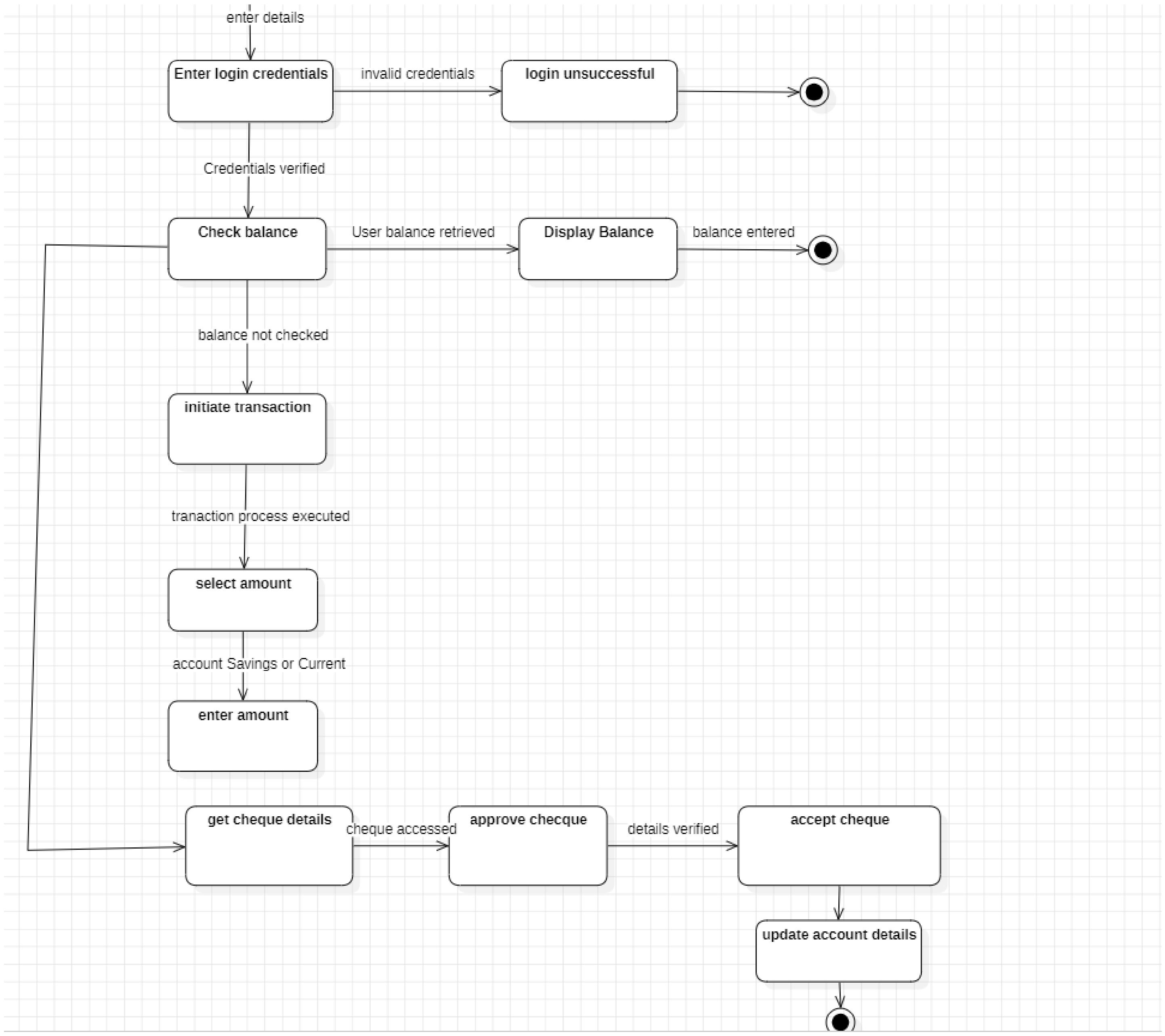
(b) Sequence Diagram:



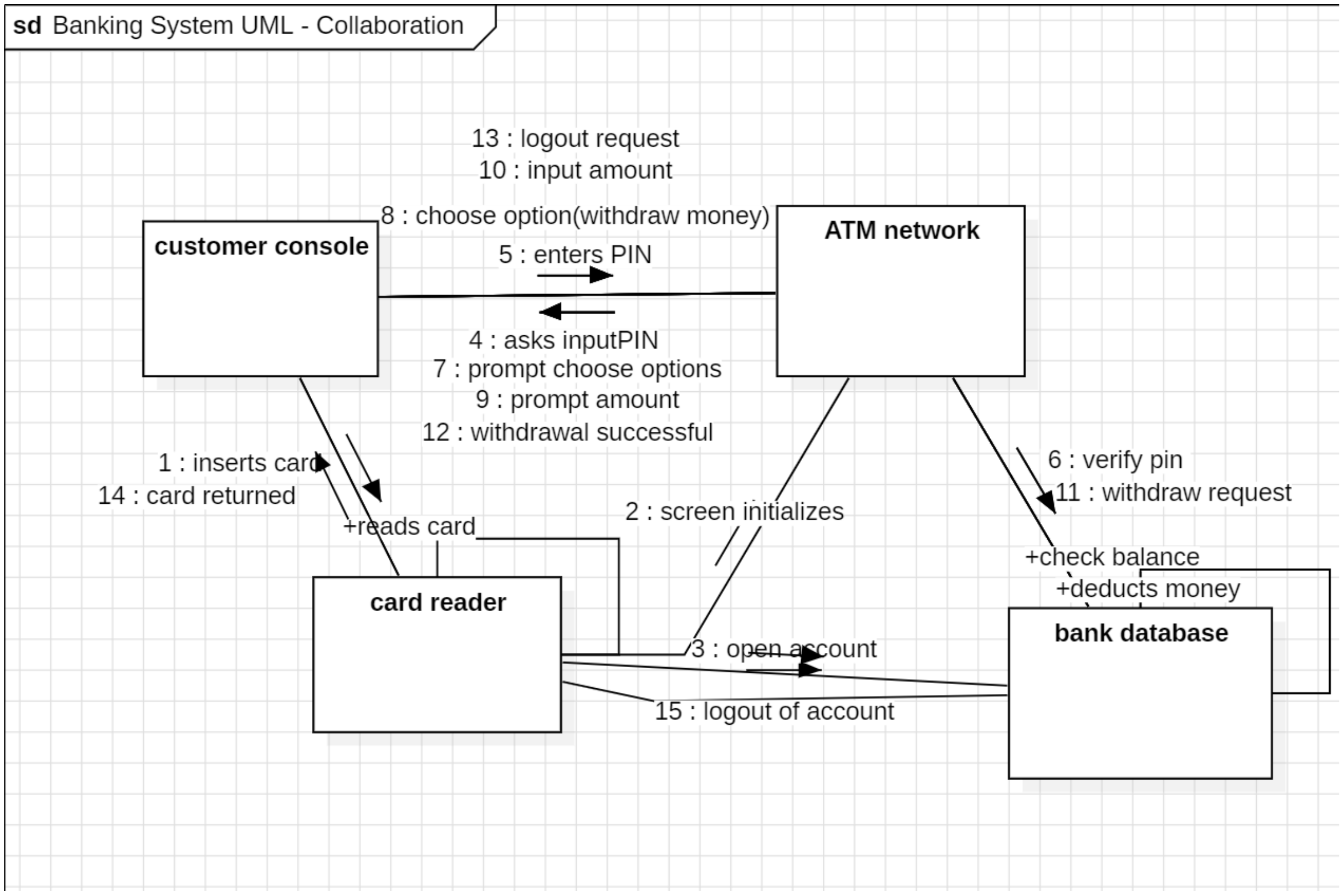
(c) Use Case Diagram:



(d) State Diagram

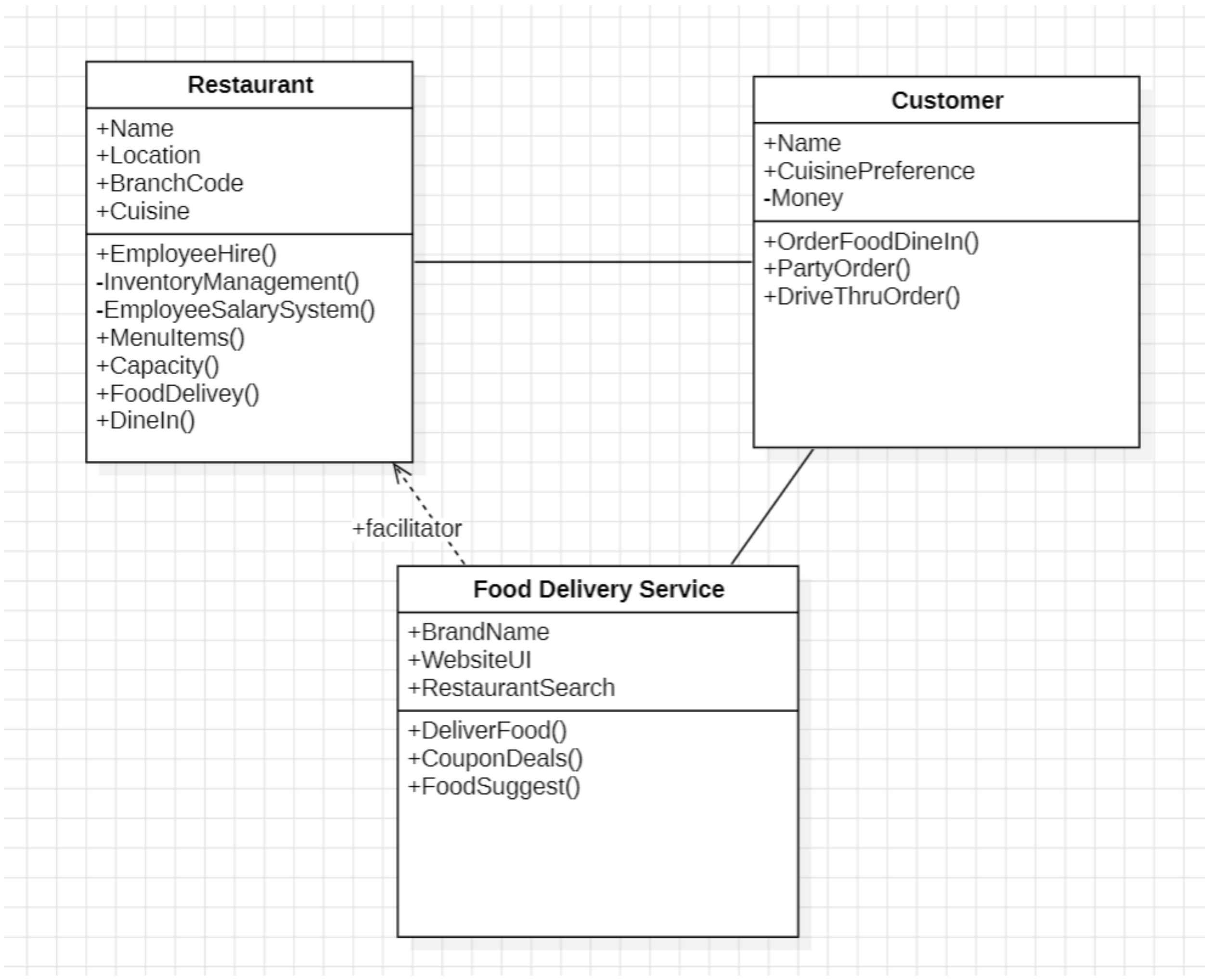


(e)Collaboration Diagram

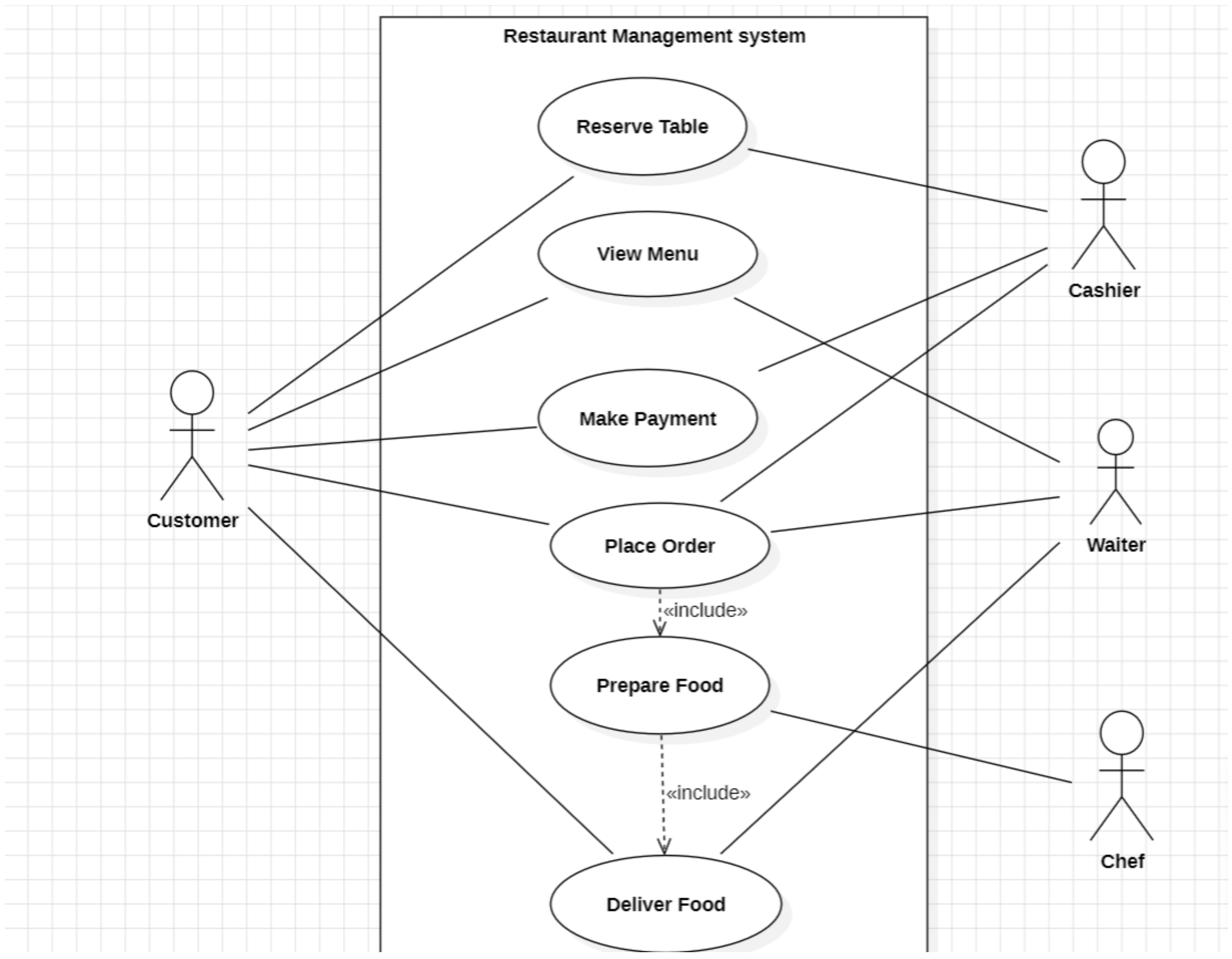


2nd UML Topic: Restaurant Management System:

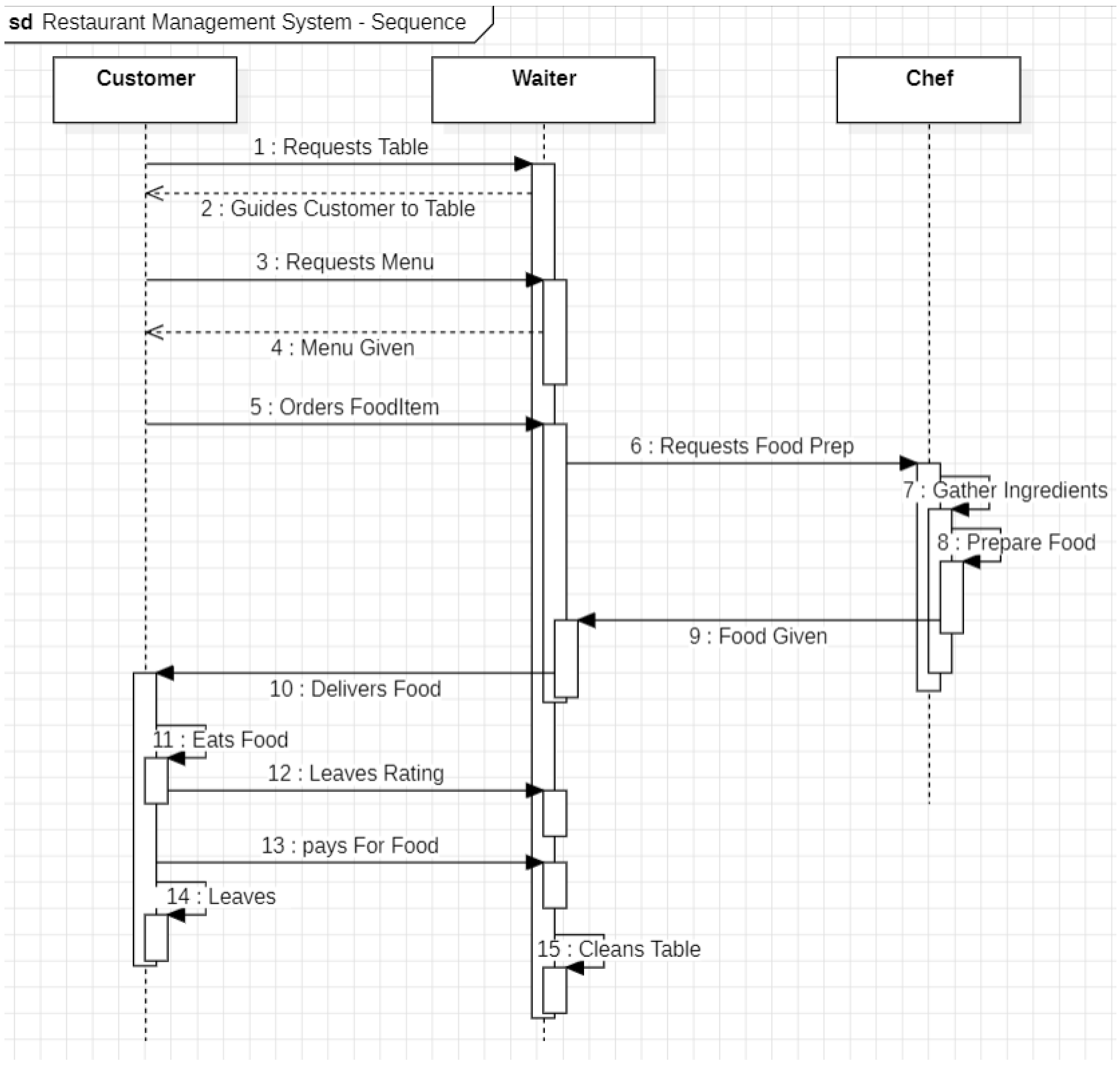
(a)Class Diagram:



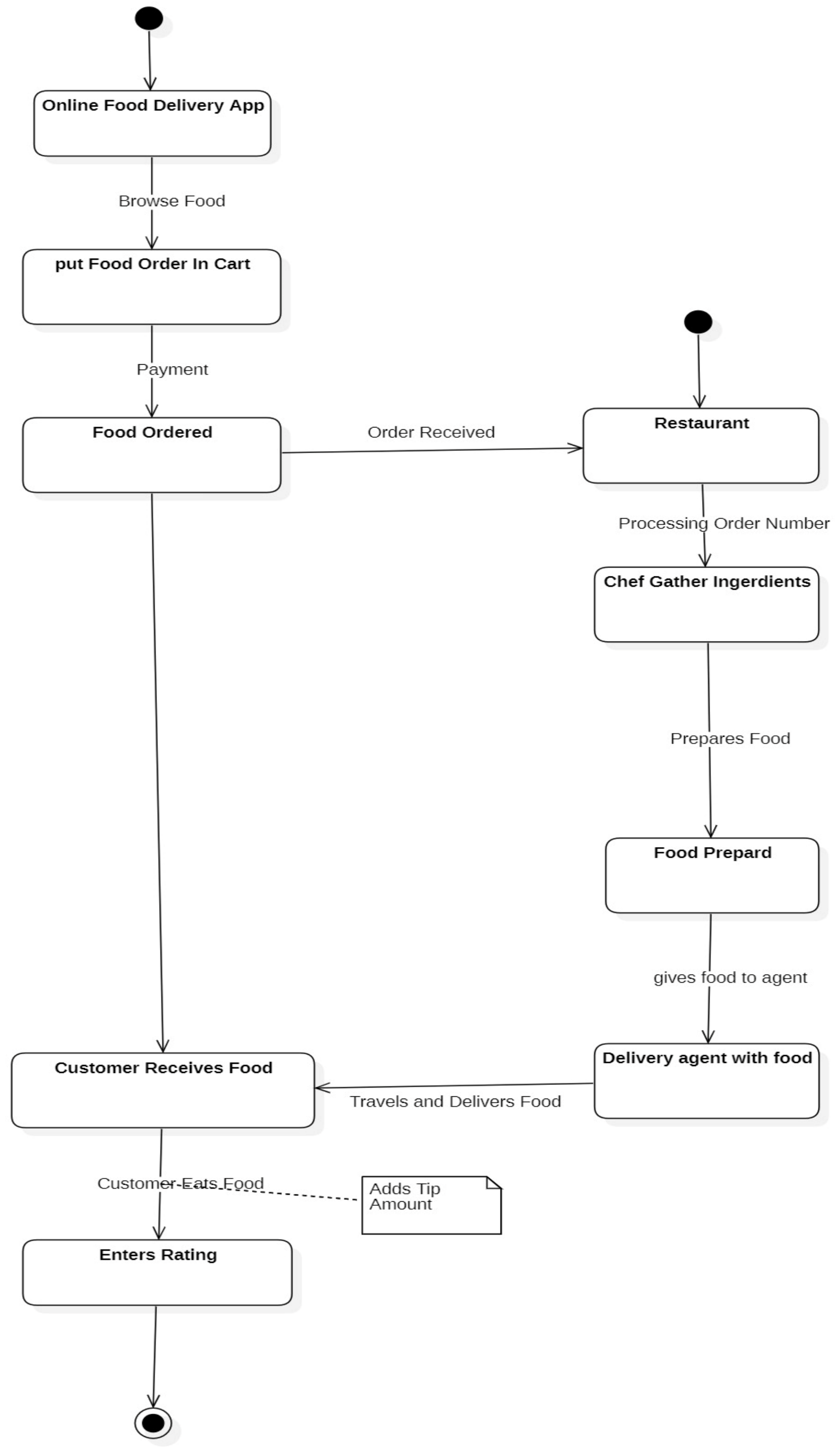
(b) Use Case Diagram:



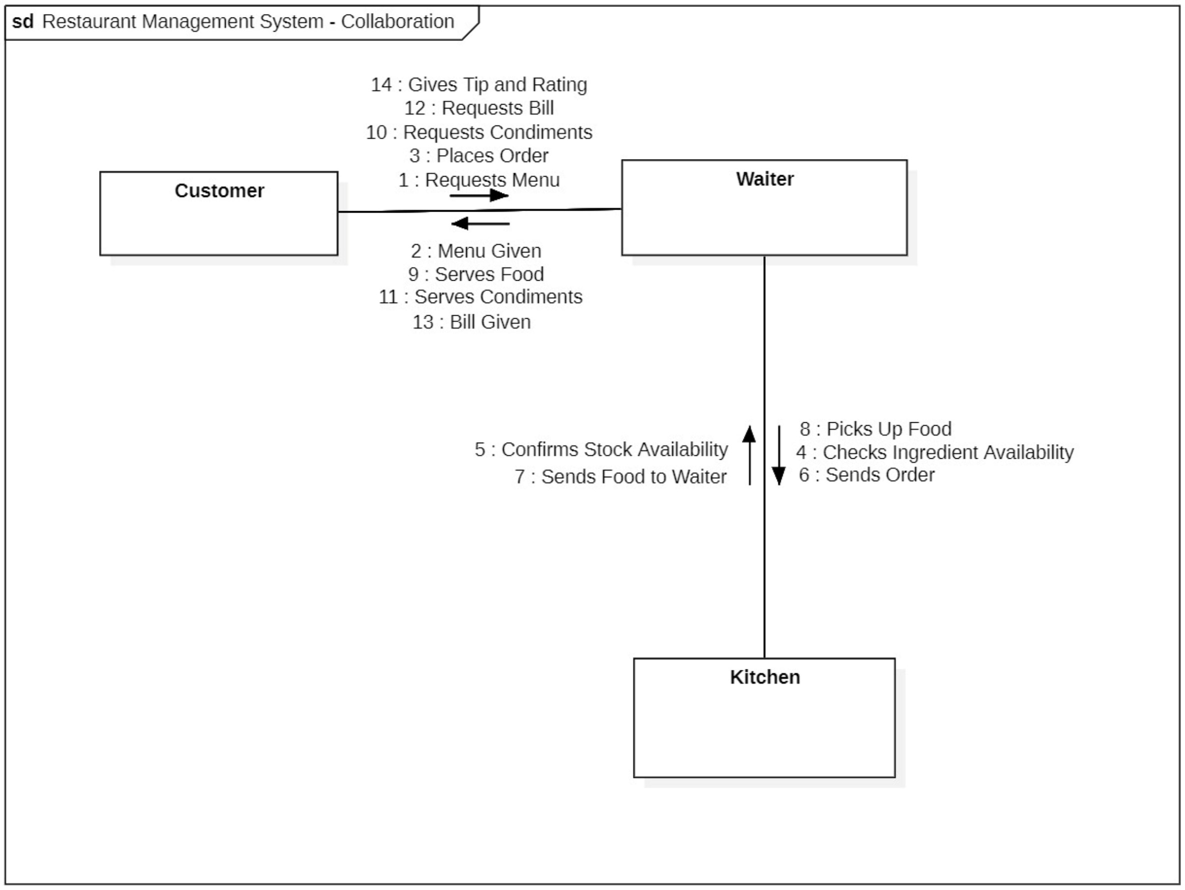
(c)Sequence Diagram:



(d) State Diagram:



(e) Collaboration Diagram:



3. Java Loop Statements, Switch Case, Jump Statements:

1. Print Numbers in Spiral Forms:

Aim: To print numbers in a **spiral-like decreasing row format** using loops and conditionally resetting the loop.

Program Code:

```
class spiralnumbers {
    public static void main(String[] args) {
        int n = 5, num = 1;
        for (int i = 0; i < n; i++) {
            System.out.print(num++ + " ");
            if (i == n - 1) {
                i = -1; n--;
                System.out.println();
            }
            if (n == 0) break;
        }
    }
}
```

Output:

```
1 2 3 4 5
 6 7 8 9
 10 11 12
 13 14
 15
```

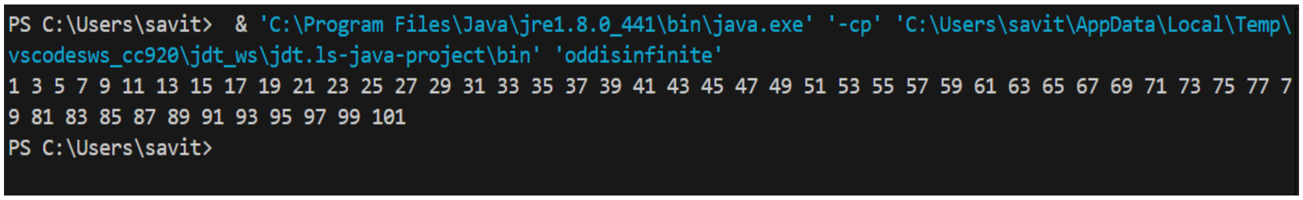
2. Infinite Odd Numbers (While Loop & Break):

Aim: To generate an **infinite odd number** and stop the loop using **break**.

Program Code:

```
class oddisinfinite {
    public static void main(String[] args) {
        int n = 1;
        while (true) {
            System.out.print(n + " ");
            if (n > 100) break;
            n += 2;
        }
    }
}
```

Output:



```
PS C:\Users\savit>  & 'C:\Program Files\Java\jre1.8.0_441\bin\java.exe' '-cp' 'C:\Users\savit\AppData\Local\Temp\
vscodesws_cc920\jdt_ws\jdt.ls-java-project\bin' 'oddisinfinite'
1 3 5 7 9 11 13 15 17 19 21 23 25 27 29 31 33 35 37 39 41 43 45 47 49 51 53 55 57 59 61 63 65 67 69 71 73 75 77 7
9 81 83 85 87 89 91 93 95 97 99 101
PS C:\Users\savit>
```

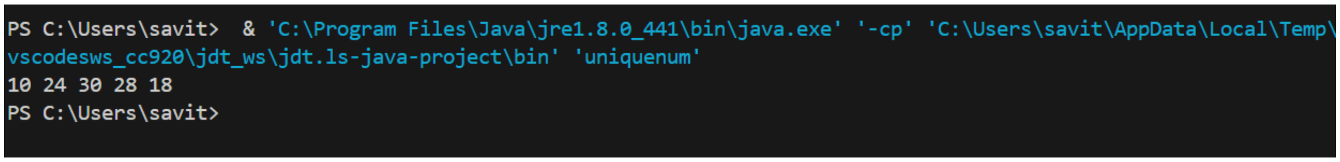
3. Unique Number Series (Loop & Continue)

Aim: To generate a **unique series of numbers** by **skipping even-indexed multiplications** using the continue statement.

Program Code:

```
class uniquenum {
    public static void main(String[] args) {
        for (int i = 1, j = 10; i <= 10; i++, j--) {
            if (i % 2 == 0) continue;
            System.out.print((i * j) + " ");
        }
    }
}
```

Output:



```
PS C:\Users\savit>  & 'C:\Program Files\Java\jre1.8.0_441\bin\java.exe' '-cp' 'C:\Users\savit\AppData\Local\Temp\
vscodesws_cc920\jdt_ws\jdt.ls-java-project\bin' 'uniquenum'
10 24 30 28 18
PS C:\Users\savit>
```

4. ATM system using switch case:

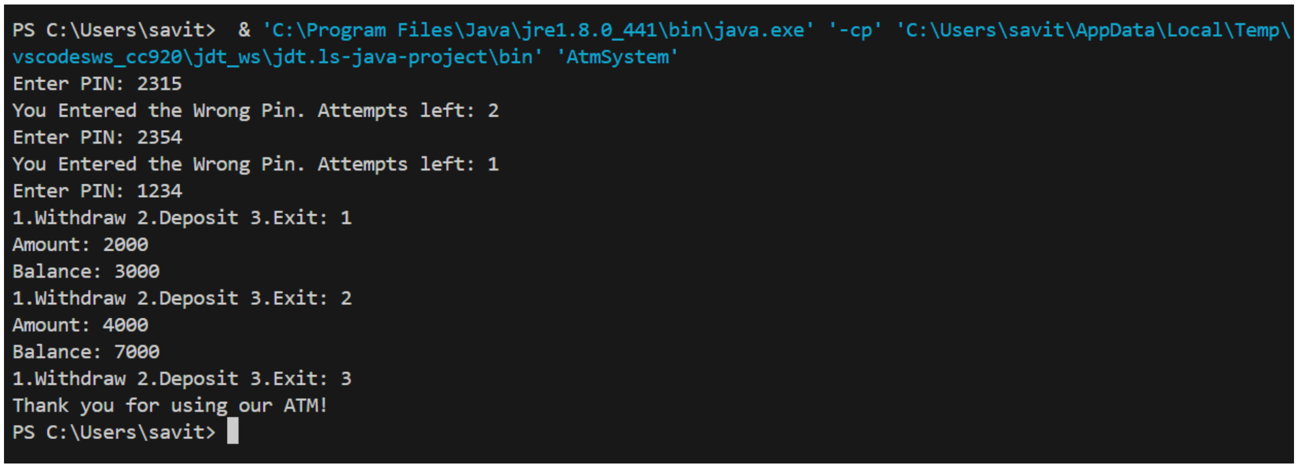
Aim: To **simulate an ATM system** that:**Validates PIN** with limited attempts and Provides options for **withdrawal, deposit, and balance check** using a switch statement

Program code:

```
import java.util.Scanner;
class Atmsystem {
    public static void main(String[] args) {
        int balance = 5000, pin = 1234;
        Scanner sc = new Scanner(System.in);
        for (int i = 3; i > 0; i--) {
            System.out.print("Enter PIN: ");
            if (sc.nextInt() == pin) break;
```

```
          System.out.println("You Entered the Wrong Pin.
Attempts left: " + (i - 1));
        }
        while (true) {
          System.out.print("1.Withdraw 2.Deposit 3.Exit: ");
          switch (sc.nextInt()) {
            case 1  { System.out.print("Amount: "); balance -=
sc.nextInt(); }
            case 2 { System.out.print("Amount: "); balance +=
sc.nextInt(); }
            case 3  System.exit(0);
          }
          System.out.println("Balance: " + balance);
        }
    }
}
```

Output:


```
PS C:\Users\savit>  & 'C:\Program Files\Java\jre1.8.0_441\bin\java.exe' '-cp' 'C:\Users\savit\AppData\Local\Temp\vscodesws_cc920\jdt_ws\jdt.ls-java-project\bin' 'AtmSystem'
Enter PIN: 2315
You Entered the Wrong Pin. Attempts left: 2
Enter PIN: 2354
You Entered the Wrong Pin. Attempts left: 1
Enter PIN: 1234
1.Withdraw 2.Deposit 3.Exit: 1
Amount: 2000
Balance: 3000
1.Withdraw 2.Deposit 3.Exit: 2
Amount: 4000
Balance: 7000
1.Withdraw 2.Deposit 3.Exit: 3
Thank you for using our ATM!
PS C:\Users\savit>
```

5. Reverse Number without String

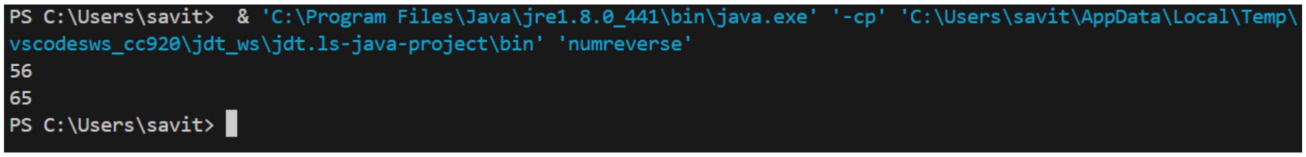
Aim: To reverse a number mathematically without converting it into a string.

Program Code:

```
import java.util.Scanner;

class numreverse {

  public static void main(String[] args) {

    int n = new Scanner(System.in).nextInt(), rev = 0;

    while (n > 0) {

      rev = rev * 10 + (n % 10);

      n /= 10;

    }

    System.out.println(rev);

  }
```

}

Output:



```
PS C:\Users\savit>  & 'C:\Program Files\Java\jre1.8.0_441\bin\java.exe' '-cp' 'C:\Users\savit\AppData\Local\Temp\vscodesws_cc920\jdt_ws\jdt.ls-java-project\bin' 'numreverse'
56
65
PS C:\Users\savit> 
```

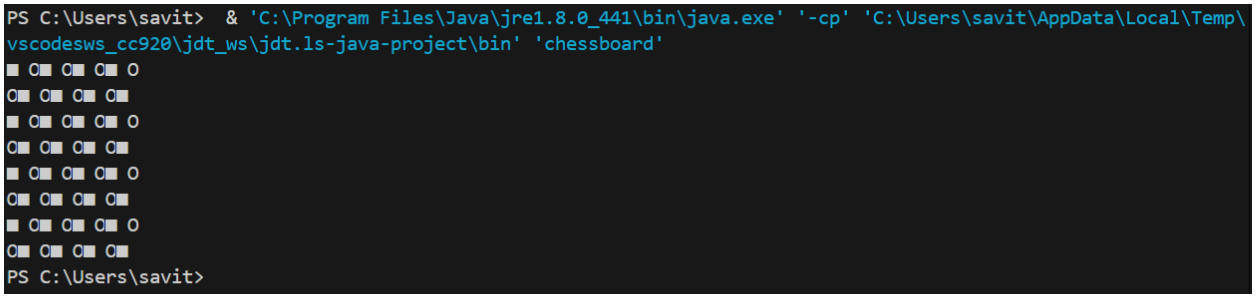
6. Print Chess Board

Aim: To print an **8x8 chessboard pattern** using **nested loops**.

Program Code:

```
class chessboard {
    public static void main(String[] args) {
        for (int i = 1; i <= 8; i++) {
            for (int j = 1; j <= 8; j++)
                System.out.print((i + j) % 2 == 0 ? "■ " : "O");
            System.out.println();
        }
    }
}
```

Output:



```
PS C:\Users\savit>  & 'C:\Program Files\Java\jre1.8.0_441\bin\java.exe' '-cp' 'C:\Users\savit\AppData\Local\Temp\
vscodesws_cc920\jdt_ws\jdt.ls-java-project\bin' 'chessboard'
■ O■ O■ O■ O
O■ O■ O■ O■
■ O■ O■ O■ O
O■ O■ O■ O■
■ O■ O■ O■ O
O■ O■ O■ O■
■ O■ O■ O■ O
O■ O■ O■ O■
PS C:\Users\savit>
```

7. Number Pyramid (For & Multiplication)

Aim: To generate **a mathematical pyramid pattern** based on **progressive number multiplication.**

Program Code:

```
class numberpyramid {
    public static void main(String[] args) {
        for (int i = 1, num = 1; i <= 5; i++, num = num * 10 + 1)
            System.out.println(num * num);
    }
}
```

Output:

```
1
121
12321
```

```
1234321

123454321
```

8. Skip multiples of 5

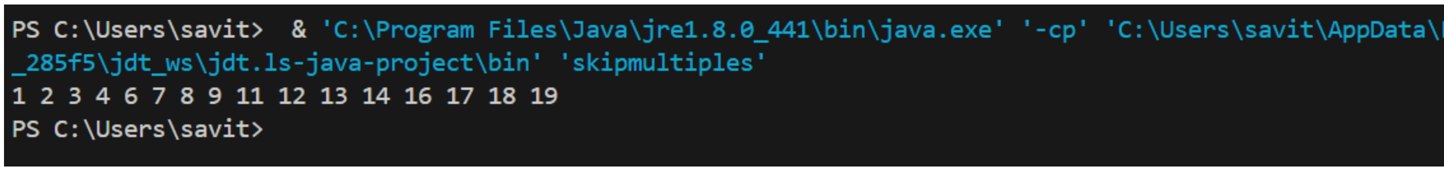
Aim: To **iterate through number series** and print the series skipping the number 5 and it multiples

Program Code:

```
class skipmultiples {
   public static void main(String[] args) {
      for (int i = 1; i <= 20; i++) {
         if (i % 5 == 0) continue;
         System.out.print(i + " ");
      }
   }
}
```

Output:

}


```
PS C:\Users\savit>  & 'C:\Program Files\Java\jre1.8.0_441\bin\java.exe' '-cp' 'C:\Users\savit\AppData\
_285f5\jdt_ws\jdt.ls-java-project\bin' 'skipmultiples'
1 2 3 4 6 7 8 9 11 12 13 14 16 17 18 19
PS C:\Users\savit>
```

1 2 3 4 6 7 8 9 11 12 13 14 16 17 18 19

9.Fibonacci Series with limit:

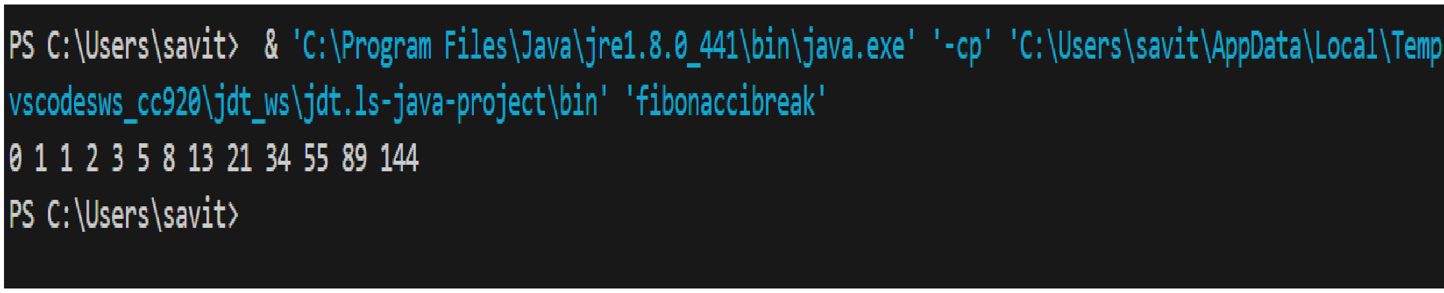
Aim: to **generate a Fibonacci series** but stop the series after a certain number

Program Code:

```
class fibonaccibreak {
  public static void main(String[] args) {
    int a = 0, b = 1;
    while (true) {
      System.out.print(a + " ");
      if (a > 100) break;
      int temp = a + b;
      a = b;
      b = temp;
    }
  }
```

}

Output:



```
PS C:\Users\savit>  & 'C:\Program Files\Java\jre1.8.0_441\bin\java.exe' '-cp' 'C:\Users\savit\AppData\Local\Temp
vscodesws_cc920\jdt_ws\jdt.ls-java-project\bin' 'fibonaccibreak'
0 1 1 2 3 5 8 13 21 34 55 89 144
PS C:\Users\savit>
```

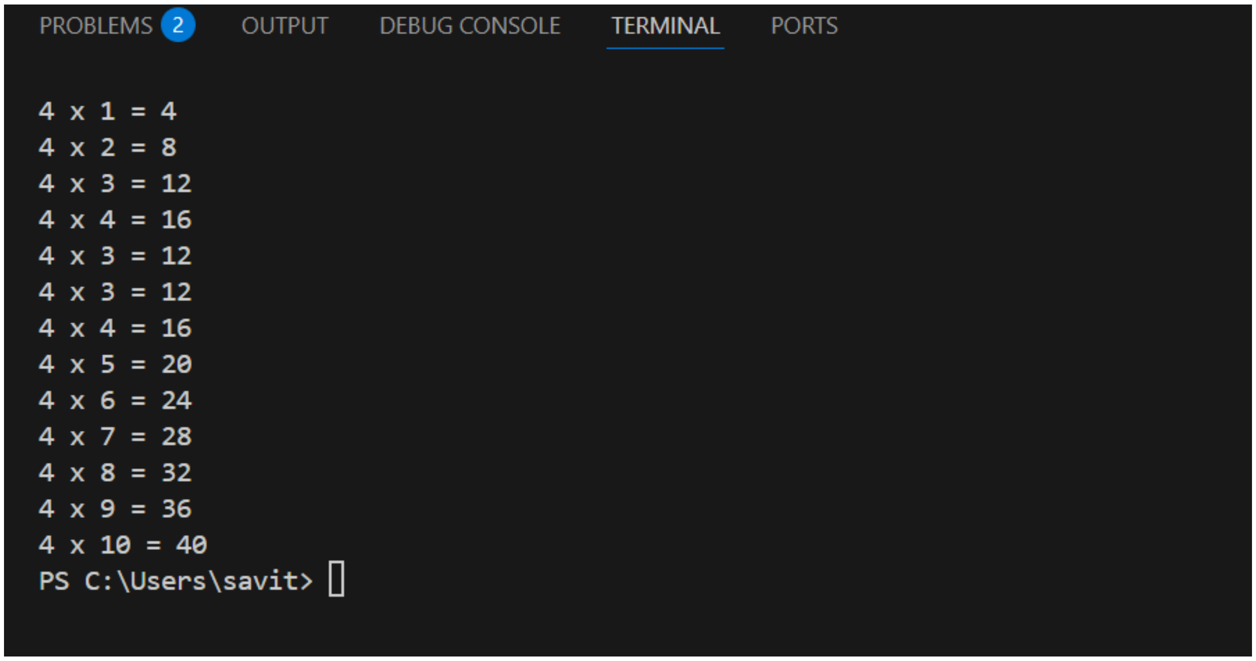
10.**Multiplication Table using for Loop:**

Aim: To generate a multiplication table with the user's number input.

Program Code:

```
import java.util.Scanner;
class multitable {
    public static void main(String[] args) {
        int n = new Scanner(System.in).nextInt();
        for (int i = 1; i <= 10; i++)
            System.out.println(n + " x " + i + " = " + (n * i));
    }
}
```

Output:



```
4 x 1 = 4
4 x 2 = 8
4 x 3 = 12
4 x 4 = 16
4 x 3 = 12
4 x 3 = 12
4 x 4 = 16
4 x 5 = 20
4 x 6 = 24
4 x 7 = 28
4 x 8 = 32
4 x 9 = 36
4 x 10 = 40
PS C:\Users\savit>
```

Inheritance:

4. Single Inheritance:

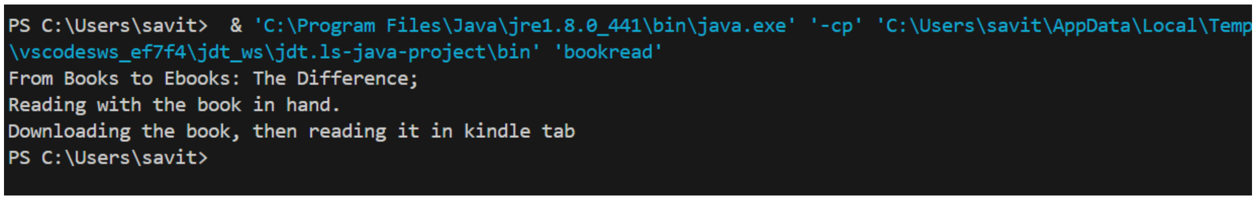
(a) Reading a book:

Aim: To show a parent class **Book** and a subclass **Ebook extending Book.** We also create two methods **read() and download()** and show that the **child class inherits the method** from parent class.

Program Code:

```java
class Book {

    void read() {
        System.out.println("Reading with the book in hand.");
    }
}

class Ebook extends Book {
    void download() {
        System.out.println("Downloading the book, then reading it in kindle tab");
    }
}

public class bookread {
    public static void main(String[] args) {
        Ebook ebook = new Ebook();
        System.out.println("From Books to Ebooks: The
Difference;");
        ebook.read();
        ebook.download();
    }
}
```

Output:



```
PS C:\Users\savit>  & 'C:\Program Files\Java\jre1.8.0_441\bin\java.exe' '-cp' 'C:\Users\savit\AppData\Local\Temp
\vscodesws_ef7f4\jdt_ws\jdt.ls-java-project\bin' 'bookread'
From Books to Ebooks: The Difference;
Reading with the book in hand.
Downloading the book, then reading it in kindle tab
PS C:\Users\savit>
```

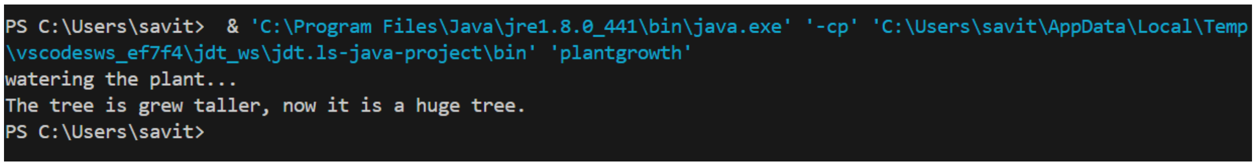
(b)Growing a Plant to a Tree:

Aim: Two classes – Plant (parent class) and Tree (child class). Tree class inherits grow() method from parent class.

Program Code:

```
class Plant {
void grow() {
    System.out.println("The tree is grew taller, now
it is a huge tree.");
    }

void water() {
    System.out.println("watering the plant...");
    }

}

class Tree extends Plant {
void grow() {
    System.out.println("The tree is grew taller, now
it is a huge tree.");
    }
}

public class plantgrowth {
public static void main(String[] args) {
    Tree tree = new Tree();
    tree.water();
    tree.grow();
    }
}
```

Output:



```
PS C:\Users\savit>  & 'C:\Program Files\Java\jre1.8.0_441\bin\java.exe' '-cp' 'C:\Users\savit\AppData\Local\Temp
\vscodesws_ef7f4\jdt_ws\jdt.ls-java-project\bin' 'plantgrowth'
watering the plant...
The tree is grew taller, now it is a huge tree.
PS C:\Users\savit>
```

5. Multilevel Inheritance Programs:

(a)Animal Evolution – From Animal to Bird to Parrot Singing:

Aim:

To have 3 classes:

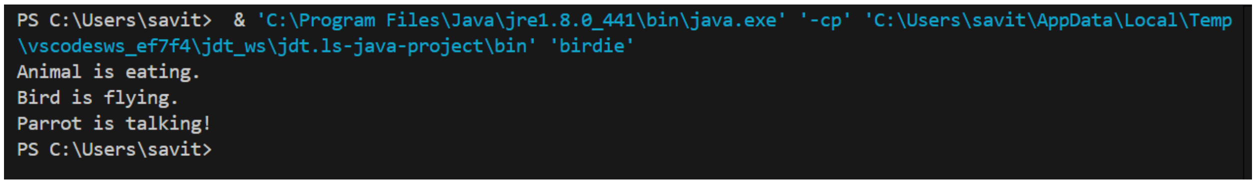
Animal, Bird and Parrot and **methods** for each class.

We aim to see if the **child class inherits methods from two classes that it extends from.**

Program Code:

```java
class Animal {
    void eat() {
        System.out.println("Animal is eating.");
    }
}

class Bird extends Animal {
    void fly() {
        System.out.println("Bird is flying.");
    }
}

class Parrot extends Bird {
    void talk() {
        System.out.println("Parrot is talking!");
    }
}

public class birdie {
    public static void main(String[] args) {
        Parrot parrot = new Parrot();
        parrot.eat();
        parrot.fly();
        parrot.talk();
    }
}
```

Output:



```
PS C:\Users\savit>  & 'C:\Program Files\Java\jre1.8.0_441\bin\java.exe' '-cp' 'C:\Users\savit\AppData\Local\Temp
\vscodesws_ef7f4\jdt_ws\jdt.ls-java-project\bin' 'birdie'
Animal is eating.
Bird is flying.
Parrot is talking!
PS C:\Users\savit>
```

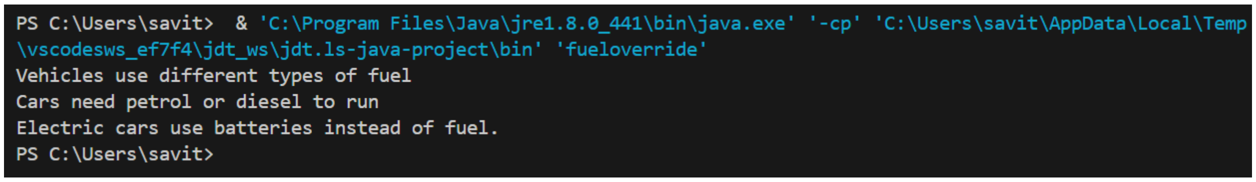
(b) Fuel difference in Car Generations:

Aim: To view **fuel types of different types of cars** through **multi-level** method deriving in java.

Program Code:

```java
class Vehicle {
    void fueltype() {
        System.out.println("Vehicles use different
types of fuel");
    }
}

class Car extends Vehicle {
    void fueltype() {
        System.out.println("Cars need petrol or
diesel to run");
    }
}

class Electriccar extends Car {
    void fueltype() {
        System.out.println("Electric cars use
batteries instead of fuel.");
    }
}

public class fueloverride {
    public static void main(String[] args) {
        Vehicle v = new Vehicle();
        v.fueltype();
        Car c = new Car();
        c.fueltype();
        Electriccar e = new Electriccar();
        e.fueltype();
    }
}
```

Output:



```
PS C:\Users\savit>  & 'C:\Program Files\Java\jre1.8.0_441\bin\java.exe' '-cp' 'C:\Users\savit\AppData\Local\Temp\vscodesws_ef7f4\jdt_ws\jdt.ls-java-project\bin' 'fueloverride'
Vehicles use different types of fuel
Cars need petrol or diesel to run
Electric cars use batteries instead of fuel.
PS C:\Users\savit>
```

6. Hierarchical Inheritance Programs:

(a)Different Delivery Options through hierarchal method in Java:

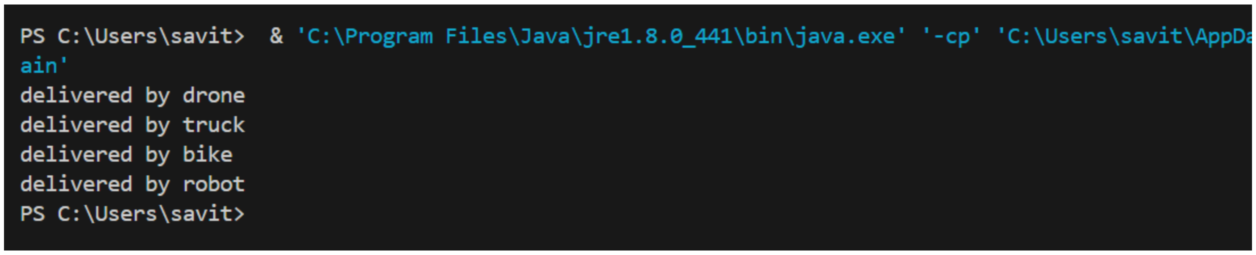
Aim: To show different delivery methods when several child classes inherit and override the parent's class method in Java.

Program Code:

```
class Delivery {
    void deliver() {
    System.out.println("delivery is on the way");
    }
}

class Drone extends Delivery {
    void deliver() {
    System.out.println("delivered by drone");
    }
}

class Truck extends Delivery {
```

```java
    void deliver() {
    System.out.println("delivered by truck");
    }
}

class Bike extends Delivery {
    void deliver() {
    System.out.println("delivered by bike");
    }
}

class Robot extends Delivery {
    void deliver() {
    System.out.println("delivered by robot");
    }
}

class Main {
    public static void main(String[] args) {
    Drone drone = new Drone();
    Truck truck = new Truck();
    Bike bike = new Bike();
    Robot robot = new Robot();
    drone.deliver();
    truck.deliver();
    bike.deliver();
    robot.deliver();
}
}
```

Output:



```
PS C:\Users\savit>  & 'C:\Program Files\Java\jre1.8.0_441\bin\java.exe' '-cp' 'C:\Users\savit\AppDa
ain'
delivered by drone
delivered by truck
delivered by bike
delivered by robot
PS C:\Users\savit>
```

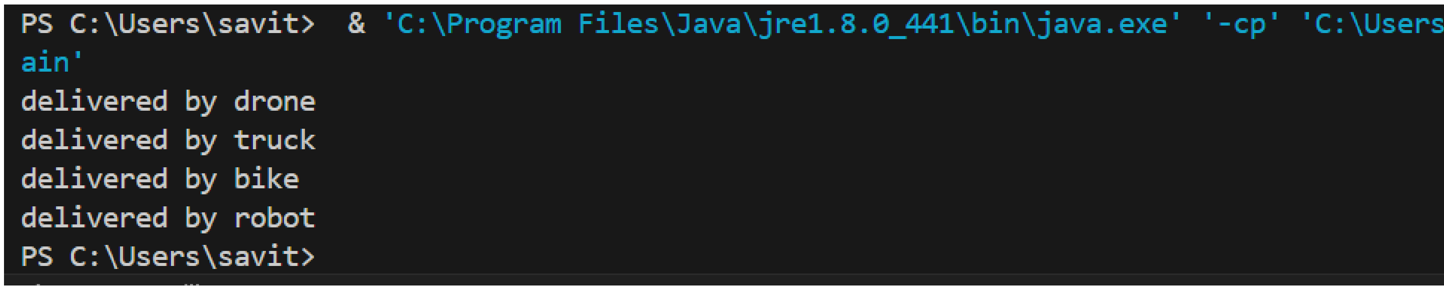
(b) Payment Options:

Aim: To show several payment options in java using derived methods from parent class.

Program Code:

```java
class Payment {
    void pay() {
        System.out.println("payment is done");
    }
}

class Card extends Payment {
    void pay() {
        System.out.println("paid using card");
    }
}

class Cash extends Payment {
    void pay() {
        System.out.println("paid using cash");
    }
}

class Main {
    public static void main(String[] args) {
        Card card = new Card();
        Cash cash = new Cash();
        card.pay();
        cash.pay();
    }
}
```

Output:



```
PS C:\Users\savit>  & 'C:\Program Files\Java\jre1.8.0_441\bin\java.exe' '-cp' 'C:\Users
ain'
delivered by drone
delivered by truck
delivered by bike
delivered by robot
PS C:\Users\savit>
```

7. Hybrid Inheritance Programs:

(a)Device Workings with Hybrid structure in Java:

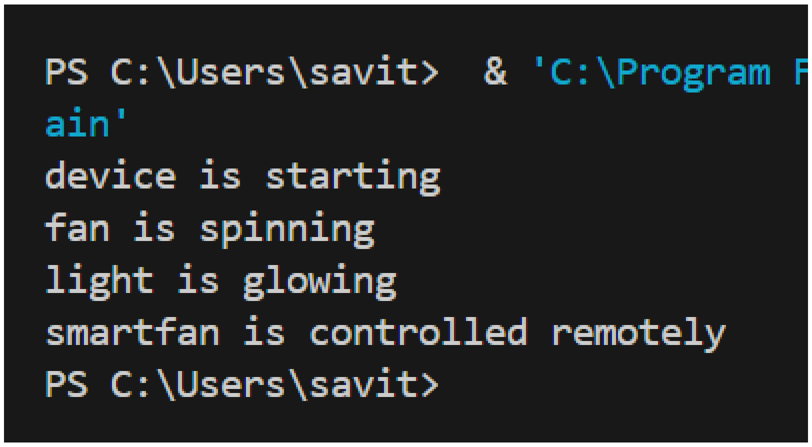
Aim: To use **Hybrid Inheritance** to observe and execute different methods related to working of different devices. A class smartfan is subclass of fan, is the single child class of a class which is derived from a mother class.

Program Code:

```java
class device {
    void start() {
    System.out.println("device is starting");
    }
}
```

```
class light extends device {
    void glow() {
    System.out.println("light is glowing");
    }
}

class fan extends device {
    void spin() {
    System.out.println("fan is spinning");
    }
}

class smartfan extends fan {
    void control() {
    System.out.println("smartfan is controlled remotely");
    }
}

class main {
    public static void main(String[] args) {
    smartfan sf = new smartfan();
    light l=new light();
    sf.start();
    sf.spin();
    l.glow();
    sf.control();
    }
}
```

Output:


```
PS C:\Users\savit>  & 'C:\Program F
ain'
device is starting
fan is spinning
light is glowing
smartfan is controlled remotely
PS C:\Users\savit>
```

(b) Decision making using **Hybrid Inheritance**:

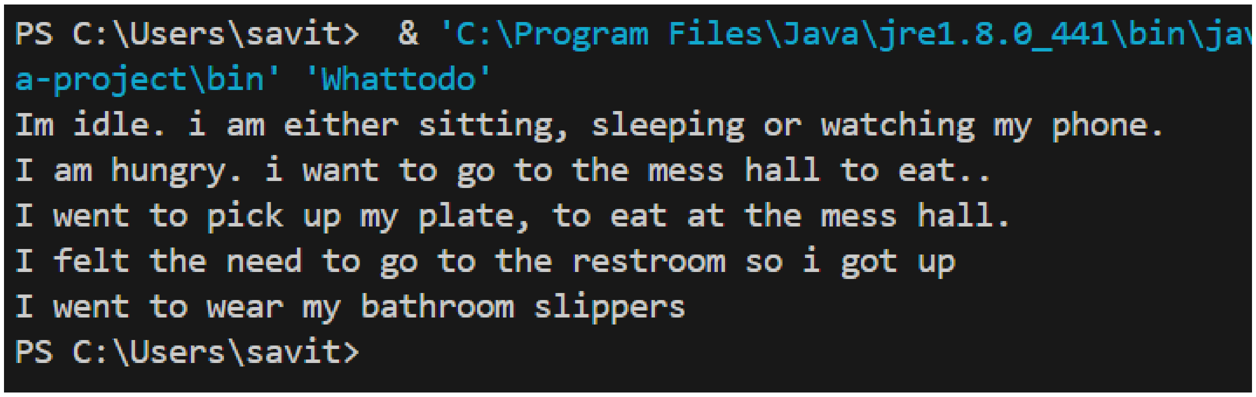
Aim: To show different decisions in a hybrid structure: how one class has a child class below it.

Program Code:

```
class idle{
    void imidle() {
        System.out.println("Im idle. i am either sitting,
sleeping or watching my phone.");
    }
}
class eating extends idle {
    void eat() {
        System.out.println("I am hungry. i want to go to the
mess hall to eat..");
    }
}
class pickupplate extends eating {
    void plate() {
        System.out.println("I went to pick up my plate, to
eat at the mess hall.");
```

```java
    }
}
class bathroom extends idle{
    void bathroom() {
        System.out.println("I felt the need to go to the restroom so i got up");
    }
}
class slippers extends bathroom {
    void slipper() {
        System.out.println("I went to wear my bathroom slippers");
    }
}
public class Whattodo{
    public static void main(String[] args){
        idle i=new idle();
        eating e=new eating();
        pickupplate p=new pickupplate();
        bathroom b=new bathroom();
        slippers s=new slippers();
        i.imidle();
        e.eat();
        p.plate();
        b.bathroom();
        s.slipper();

    }
}
```

Output:



```
PS C:\Users\savit>  & 'C:\Program Files\Java\jre1.8.0_441\bin\ja
a-project\bin' 'Whattodo'
Im idle. i am either sitting, sleeping or watching my phone.
I am hungry. i want to go to the mess hall to eat..
I went to pick up my plate, to eat at the mess hall.
I felt the need to go to the restroom so i got up
I went to wear my bathroom slippers
PS C:\Users\savit>
```

Polymorphism

8. Constructor Polymorphism:

(a) Product Purchase:

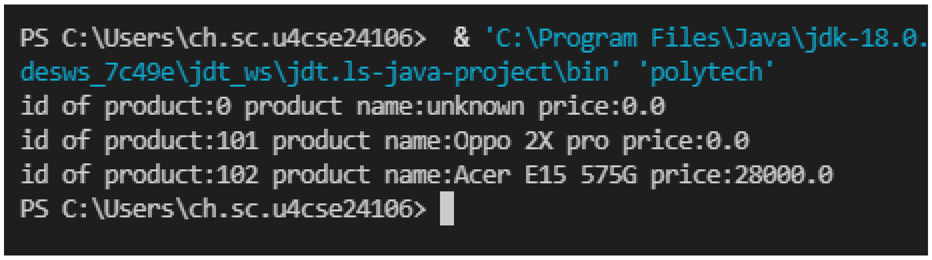
Aim:

To implement **constructor polymorphism** in Java by overloading constructors in a product class, allowing objects to be created with different levels of information. This demonstrates how multiple constructors enable flexible object initialization.

Program Code:

```java
class product {
    void display() {
    System.out.println("id of product:"+id+" product
name:"+name+" price:"+price);
        }
    int id;
    String name;
    double price;
product() {
    id=0;
    name="unknown";
    price=0.0;
}
product(int i,String n) {
    id=i;
    name=n;
    price=0.;
}
product(int i,String n,double p) {
    id=i;
    name=n;
    price=p;
}
}
public class polytech {
public static void main(String[] args) {
    product p1=new product();
    product p2=new product(101,"Oppo 2X pro");
    product p3=new product(102,"Acer E15 575G",28000);
p1.display();
p2.display();
p3.display();
}
}
```

Output:



```
PS C:\Users\ch.sc.u4cse24106>  & 'C:\Program Files\Java\jdk-18.0.
desws_7c49e\jdt_ws\jdt.ls-java-project\bin' 'polytech'
id of product:0 product name:unknown price:0.0
id of product:101 product name:Oppo 2X pro price:0.0
id of product:102 product name:Acer E15 575G price:28000.0
PS C:\Users\ch.sc.u4cse24106>
```

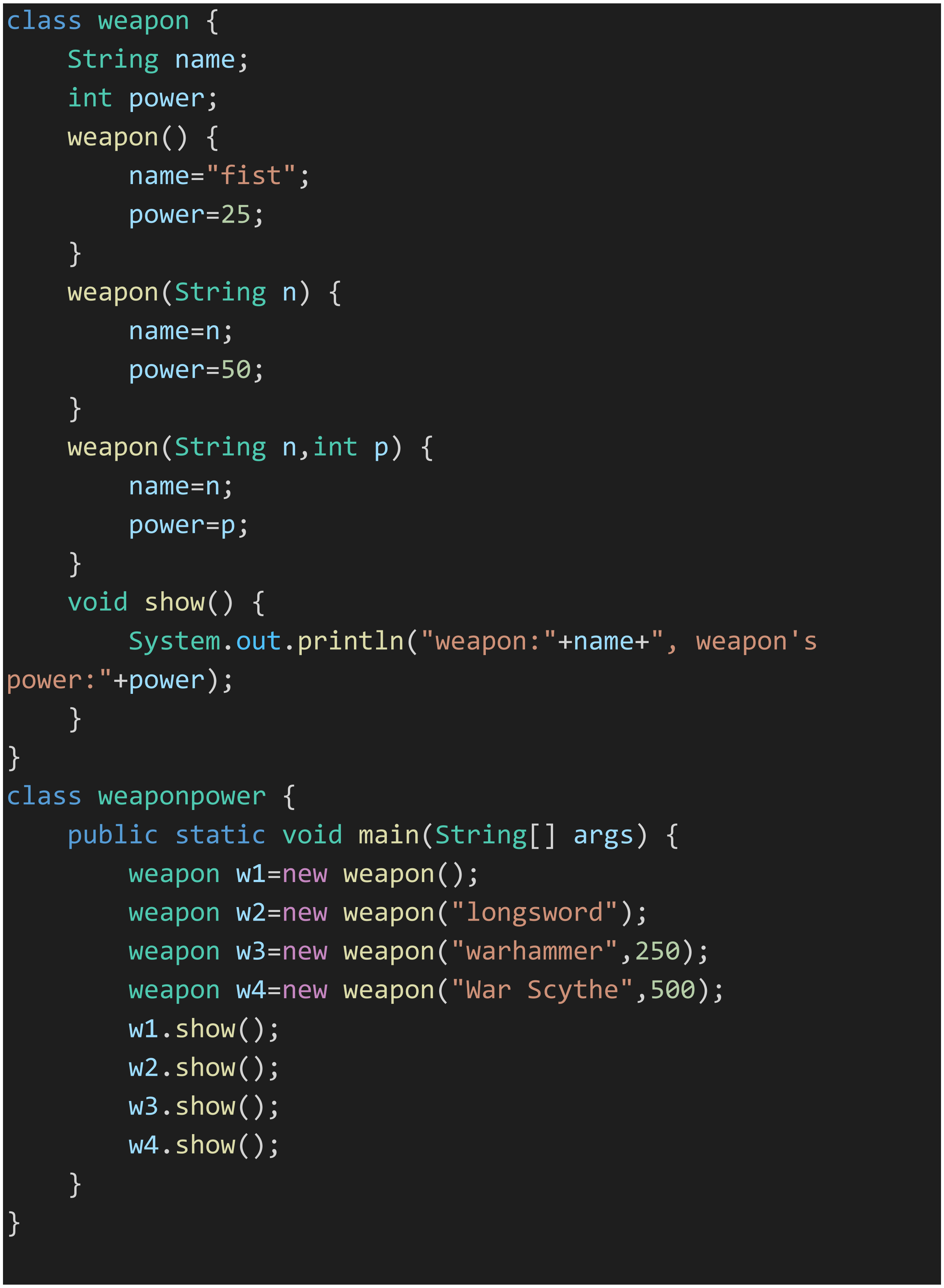
9. Constructor **Overloading** Programs:

(a) Implementation of Constructor Polymorphism in a **Weapon System**

Aim: To demonstrate **constructor polymorphism** in Java by overloading constructors in a weapon class, allowing weapons to be created with different names and power levels.

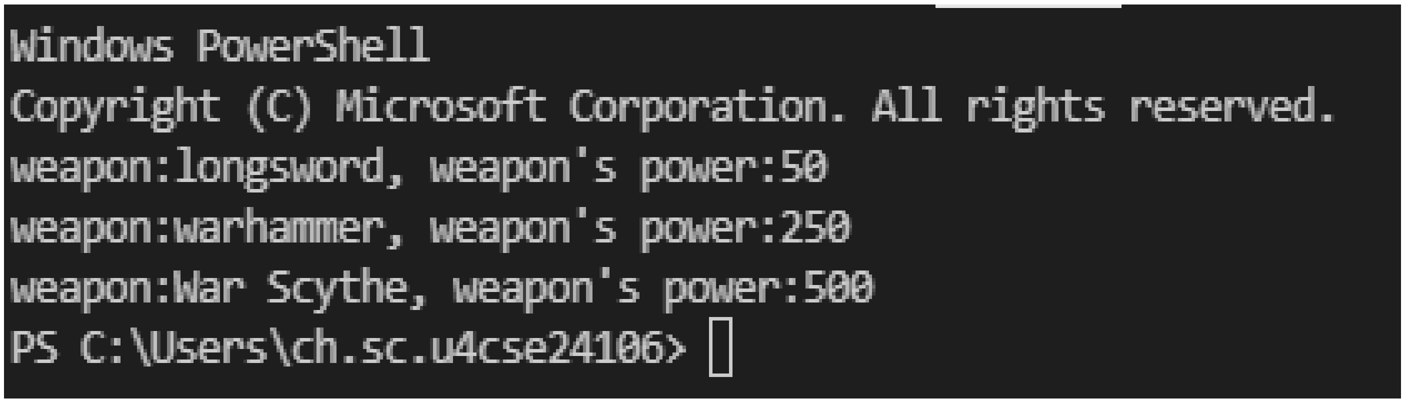
This showcases how multiple constructors provide flexibility in object initialization.

Program Code:



```java
class weapon {
    String name;
    int power;
    weapon() {
        name="fist";
        power=25;
    }
    weapon(String n) {
        name=n;
        power=50;
    }
    weapon(String n,int p) {
        name=n;
        power=p;
    }
    void show() {
        System.out.println("weapon:"+name+", weapon's 
power:"+power);
    }
}
class weaponpower {
    public static void main(String[] args) {
        weapon w1=new weapon();
        weapon w2=new weapon("longsword");
        weapon w3=new weapon("warhammer",250);
        weapon w4=new weapon("War Scythe",500);
        w1.show();
        w2.show();
        w3.show();
        w4.show();
    }
}
```

Output:



```
Windows PowerShell
Copyright (C) Microsoft Corporation. All rights reserved.
weapon:longsword, weapon's power:50
weapon:warhammer, weapon's power:250
weapon:War Scythe, weapon's power:500
PS C:\Users\ch.sc.u4cse24106>
```

10. Method **Overloading** Programs:

(a) Juice Customization System:

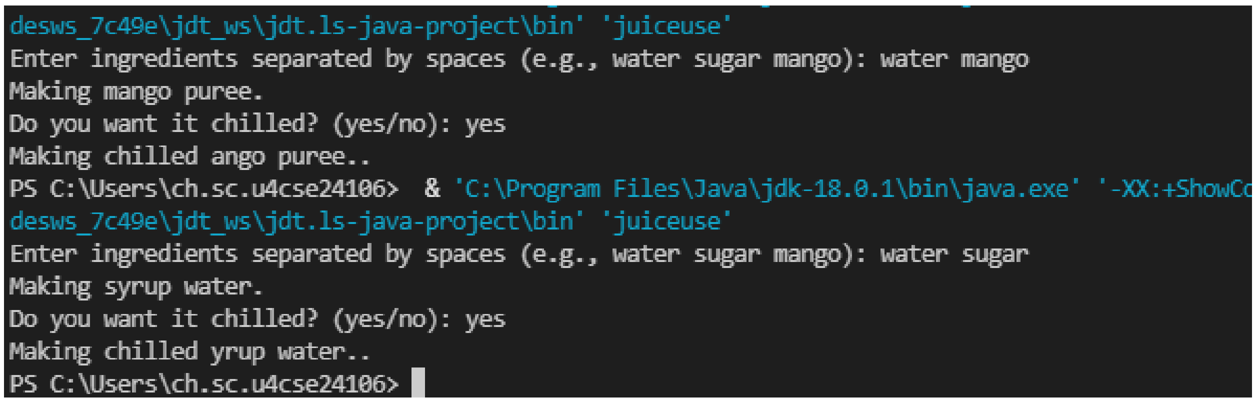
Aim:

To demonstrate **method overloading and method overriding** in Java by creating a juice-making system where different ingredient combinations produce different drinks, and an extended class provides specialized options like chilled drinks.

Program Code:

```java
import java.util.Scanner;

class juice {
    String make(String... ingredients) {
        if (ingredients.length < 2) return "At least 2 ingredients are necessary.";
        boolean haswater = false, hassugar = false, hasmango = false, hasmilk = false, hasice = false;
        for (String ingredient : ingredients) {
            switch (ingredient.toLowerCase()) {
                case "water": haswater = true; break;
                case "sugar": hassugar = true; break;
                case "mango": hasmango = true; break;
                case "milk": hasmilk = true; break;
                case "ice": hasice = true; break;
            }
        }
        if (hasmilk && hasmango) return "Making a mango milkshake.";
        if (haswater && hassugar && !hasmango) return "Making syrup water.";
        if (haswater && hassugar && hasmango) return "Making mango juice.";
        if (haswater && hasmango && hasice && !hassugar) return "Making sweetless mango juice.";
        if (haswater && hasmango && !hassugar && !hasice) return "Making mango puree.";
        return "Invalid ingredient combination.";
    }
}

class specializedjuice extends juice {
    String make(String drink, boolean addice) {
        return addice ? "Making chilled " + drink + "." :
"Making " + drink + ".";
    }
}

public class juiceuse {
    public static void main(String[] args) {
        Scanner u = new Scanner(System.in);
        System.out.print("Enter ingredients separated by
spaces (e.g., water sugar mango): ");
        String input = u.nextLine();
        String[] ingredients = input.split(" ");
        juice j = new juice();
        String result = j.make(ingredients);
        System.out.println(result);
        if (!result.contains("Invalid") &&
!result.contains("At least")) {
            System.out.print("Do you want it chilled?
(yes/no): ");
            String choice = u.next();
            boolean addice = choice.equalsIgnoreCase("yes");
            specializedjuice sj = new specializedjuice();
            System.out.println(sj.make(result.substring(8),
addice));
        }
    }
}
```

Output:



```
desws_7c49e\jdt_ws\jdt.ls-java-project\bin' 'juiceuse'
Enter ingredients separated by spaces (e.g., water sugar mango): water mango
Making mango puree.
Do you want it chilled? (yes/no): yes
Making chilled ango puree..
PS C:\Users\ch.sc.u4cse24106>  & 'C:\Program Files\Java\jdk-18.0.1\bin\java.exe' '-XX:+ShowCo
desws_7c49e\jdt_ws\jdt.ls-java-project\bin' 'juiceuse'
Enter ingredients separated by spaces (e.g., water sugar mango): water sugar
Making syrup water.
Do you want it chilled? (yes/no): yes
Making chilled yrup water..
PS C:\Users\ch.sc.u4cse24106> 
```

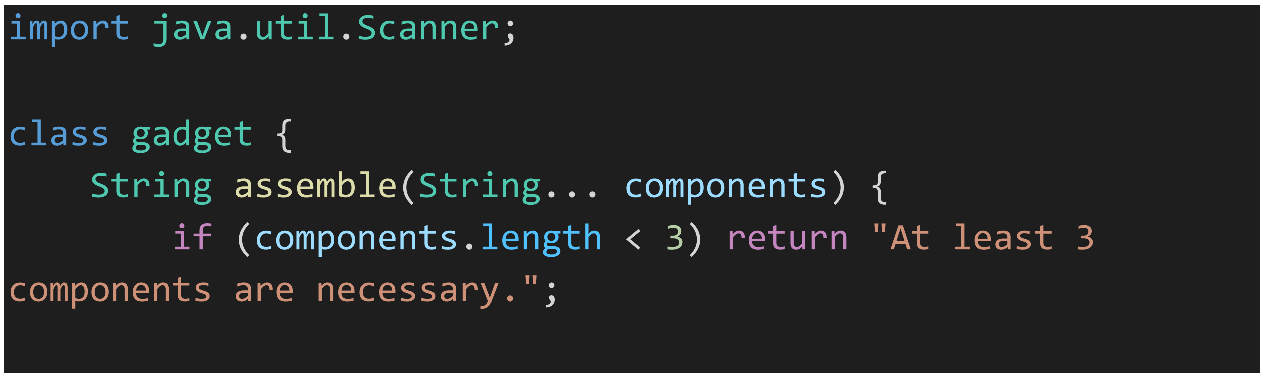
(b) Robot Assembly System

Aim:

To create a **robot assembly system** where users select components, and the program determines the type of robot being built.

The system uses **method overloading and inheritance** to allow both **basic** and **specialized** robots to be assembled and upgraded.

Program Code:

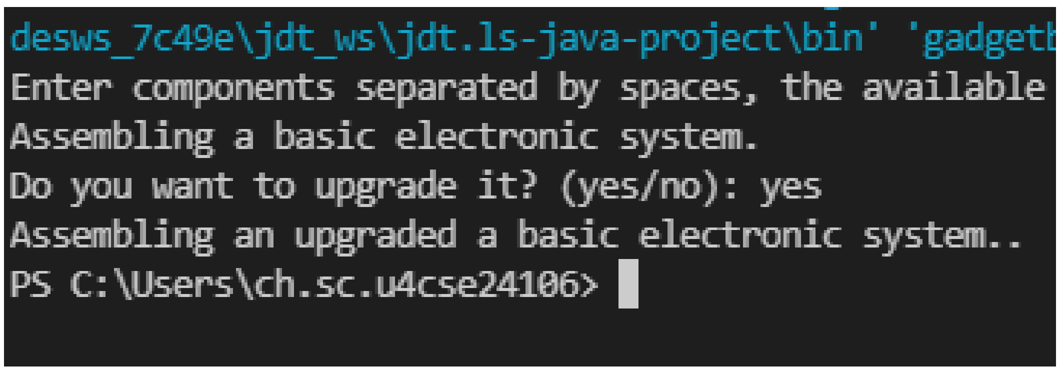


```
import java.util.Scanner;

class gadget {
    String assemble(String... components) {
        if (components.length < 3) return "At least 3 
components are necessary.";
```

```
        boolean hascircuit = false, hasbattery = false,
hassensor = false, hasmotor = false;
        boolean hasaiunit = false, haswheels = false,
hasframe = false;

        for (String component : components) {
            component = component.toLowerCase();
            if (component.equals("circuit")) hascircuit =
true;
            else if (component.equals("battery")) hasbattery
= true;
            else if (component.equals("sensor")) hassensor =
true;
            else if (component.equals("motor")) hasmotor =
true;
            else if (component.equals("aiunit")) hasaiunit =
true;
            else if (component.equals("wheels")) haswheels =
true;
            else if (component.equals("frame")) hasframe =
true;
        }

        if (hascircuit && hasbattery && hasframe) return
"Assembling a basic electronic system.";
        if (hascircuit && hasbattery && hassensor) return
"Assembling a smart sensor module.";
        if (hascircuit && hasbattery && hasmotor &&
hasframe) return "Assembling a simple robot.";
        if (hascircuit && hasbattery && hasmotor &&
haswheels && hasframe) return "Assembling a mobile robot.";
```

```
        if (hascircuit && hasbattery && hasmotor &&
hasaiunit && hasframe) return "Assembling an autonomous
robot.";
        if (hascircuit && hasbattery && hasmotor &&
haswheels && hasaiunit && hasframe) return "Assembling an
AI-powered mobile robot.";

        return "Invalid component combination.";
    }
}

class specializedgadget extends gadget {
    String assemble(String gadget, boolean upgrade) {
        return upgrade ? "Assembling an upgraded " + gadget
+ "." : "Assembling a standard " + gadget + ".";
    }
}

public class gadgetbuilder {
    public static void main(String[] args) {
        Scanner scanner = new Scanner(System.in);

        System.out.print("Enter components separated by
spaces, the available components are circuit, battery,
sensor, ai unit, motor, wheels and frame: ");
        String input = scanner.nextLine();
        String[] components = input.split(" ");

        gadget g = new gadget();
        String result = g.assemble(components);
        System.out.println(result);
        if (!result.contains("Invalid") &&
!result.contains("At least")) {
            System.out.print("Do you want to upgrade it?
(yes/no): ");
            String choice = scanner.next();
            boolean upgrade =
choice.equalsIgnoreCase("yes");

            specializedgadget sg = new specializedgadget();

System.out.println(sg.assemble(result.substring(11),
upgrade));
        }
    }
}
```

Output:



```
desws_7c49e\jdt_ws\jdt.ls-java-project\bin' 'gadgetb
Enter components separated by spaces, the available
Assembling a basic electronic system.
Do you want to upgrade it? (yes/no): yes
Assembling an upgraded a basic electronic system..
PS C:\Users\ch.sc.u4cse24106>
```

11. Method Overriding Programs:

(a) Flour Usage System

Aim: This program demonstrates the concept of **inheritance and method overriding** in Java. The base class flour stores the amount of flour available, while the **derived classes bread, cake, and pasta inherit from flour and override the use method** to specify the required flour for each food item.

The program takes **user input f**or the initial flour amount and simulates the process of making these items while updating the remaining flour.

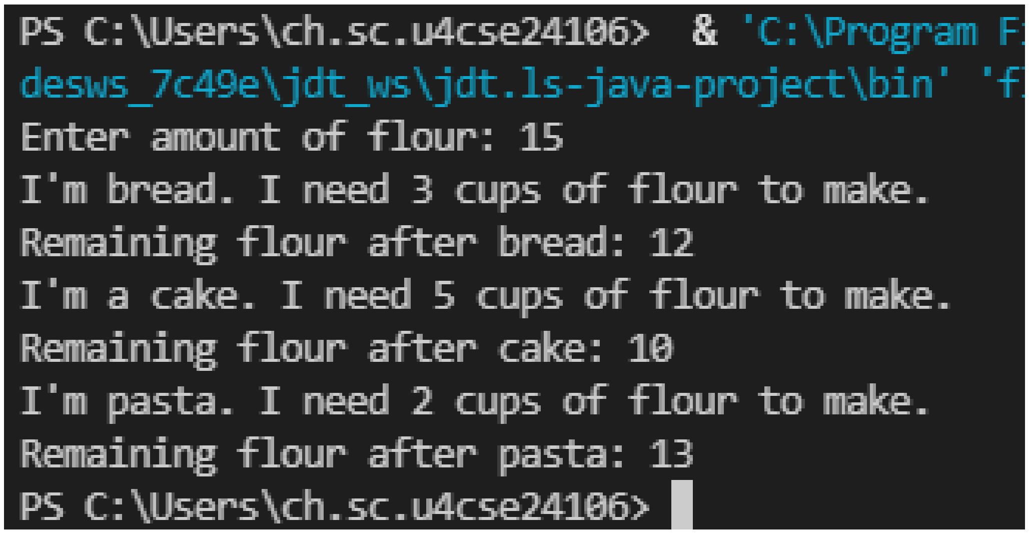
Program Code:

```
import java.util.Scanner;

class flour {
    int flouramt;
    flour(int flouramt) {
        this.flouramt = flouramt;
    }
    void use() {
        System.out.println("I'm flour. I am used to make 
different kinds of food.");
    }
    int flourer() {
```

```java
        return flouramt;
    }
}

class bread extends flour {
    bread(int flouramt) {
        super(flouramt);
    }
    void use() {
        if (flouramt < 3) {
            System.out.println("flour amount insufficient");
        } else {
            System.out.println("I'm bread. I need 3 cups of
flour to make.");
            flouramt -= 3;
        }
    }
}

class cake extends flour {
    cake(int flouramt) {
        super(flouramt);
    }
    void use() {
        if (flouramt < 5) {
            System.out.println("flour amount insufficient");
        } else {
            System.out.println("I'm a cake. I need 5 cups of
flour to make.");
            flouramt -= 5;
        }
    }
}
```

```java
class pasta extends flour {
    pasta(int flouramt) {
        super(flouramt);
    }
    void use() {
        if (flouramt < 2) {
            System.out.println("flour amount insufficient");
        } else {
            System.out.println("I'm pasta. I need 2 cups of
flour to make.");
            flouramt -= 2;
        }
    }
}
public class flourcook300 {
    public static void main(String[] args) {
        Scanner u = new Scanner(System.in);
        System.out.print("Enter amount of flour: ");
        int flouramt = u.nextInt();
        bread b = new bread(flouramt);
        cake c = new cake(flouramt);
        pasta p = new pasta(flouramt);
        b.use();
        System.out.println("Remaining flour after bread: " +
b.flourer());
        c.use();
        System.out.println("Remaining flour after cake: " +
c.flourer());
        p.use();
        System.out.println("Remaining flour after pasta: " +
p.flourer());
    }
}
```

Output:



```
PS C:\Users\ch.sc.u4cse24106>  & 'C:\Program F
desws_7c49e\jdt_ws\jdt.ls-java-project\bin' 'f
Enter amount of flour: 15
I'm bread. I need 3 cups of flour to make.
Remaining flour after bread: 12
I'm a cake. I need 5 cups of flour to make.
Remaining flour after cake: 10
I'm pasta. I need 2 cups of flour to make.
Remaining flour after pasta: 13
PS C:\Users\ch.sc.u4cse24106>
```

(b) Steel Usage simulator:

Aim: to simulate how **steel is consumed** in manufacturing various objects like **chairs, bottles, and car frames**, using **inheritance and method overriding** to represent different steel-based items.

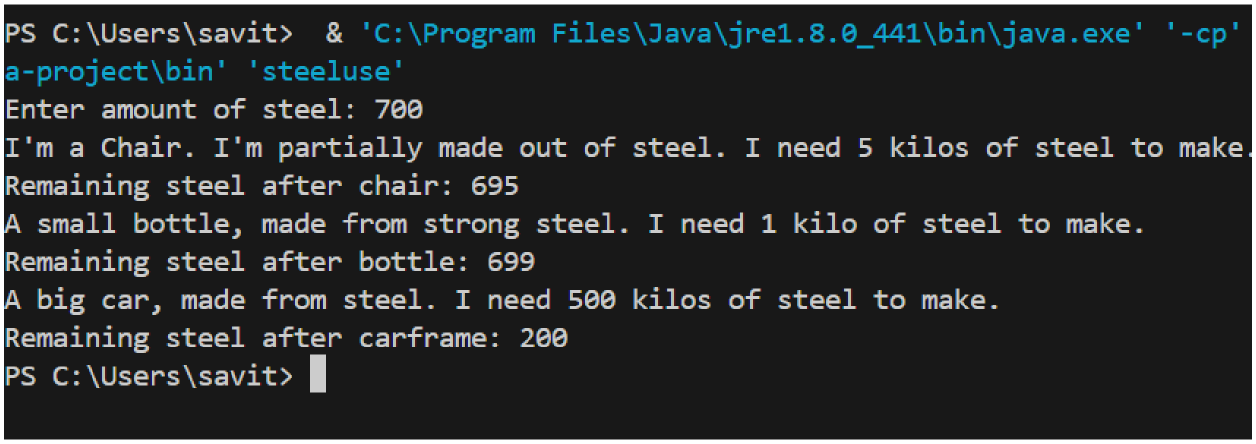
Program Code:

```
import java.util.Scanner;

class steel {
    int steelamt;
    steel(int steelamt) {
```

```java
        this.steelamt = steelamt;
    }
    void use() {
        System.out.println("I'm steel. I am used to make other useful materials.");
    }
    int steeler() {
        return steelamt;
    }
}

class chair extends steel {
    chair(int steelamt) {
        super(steelamt);
    }
    void use() {
        if (steelamt < 5) {
            System.out.println("steel amount insufficient");
        } else {
            System.out.println("I'm a Chair. I'm partially made out of steel. I need 5 kilos of steel to make.");
            steelamt -= 5;
        }
    }
}

class bottle extends steel {
    bottle(int steelamt) {
        super(steelamt);
    }
    void use() {
        if (steelamt < 1) {
            System.out.println("steel amount insufficient");
        } else {
            System.out.println("A small bottle, made from strong steel. I need 1 kilo of steel to make.");
            steelamt -= 1;
        }
    }
```

```
}

class carframe extends steel {
    carframe(int steelamt) {
        super(steelamt);
    }
    void use() {
        if (steelamt < 500) {
            System.out.println("steel amount insufficient");
        } else {
            System.out.println("A big car, made from steel.
I need 500 kilos of steel to make.");
            steelamt -= 500;
        }
    }
}

public class steeluse {
    public static void main(String[] args) {
        Scanner u = new Scanner(System.in);
        System.out.print("Enter amount of steel: ");
        int steelamt = u.nextInt();
        chair c = new chair(steelamt);
        bottle b = new bottle(steelamt);
        carframe cf = new carframe(steelamt);
        c.use();
        System.out.println("Remaining steel after chair: " +
c.steeler());
        b.use();
        System.out.println("Remaining steel after bottle: "
+ b.steeler());
        cf.use();
        System.out.println("Remaining steel after carframe:
" + cf.steeler());
    }
}
```

Output:



```
PS C:\Users\savit>  & 'C:\Program Files\Java\jre1.8.0_441\bin\java.exe' '-cp'
a-project\bin' 'steeluse'
Enter amount of steel: 700
I'm a Chair. I'm partially made out of steel. I need 5 kilos of steel to make.
Remaining steel after chair: 695
A small bottle, made from strong steel. I need 1 kilo of steel to make.
Remaining steel after bottle: 699
A big car, made from steel. I need 500 kilos of steel to make.
Remaining steel after carframe: 200
PS C:\Users\savit>
```

Abstraction:

12. Interface Programs:

(a) Dynamic Shape Transformer

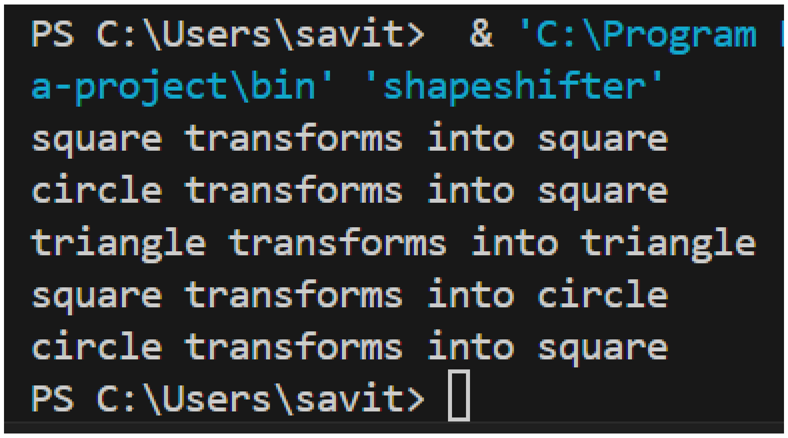
Aim: Demonstrates **interfaces and polymorphism** by defining a shape interface that multiple classes (circle, square, triangle) implement. Each class provides its own version of the transform method, allowing different behaviours while following a common structure.

Program Code:

```java
import java.util.Random;

interface shape {
    void transform();
}

class circle implements shape {
    public void transform() {
        System.out.println("circle transforms into "
+ shapeshifter.getrandomshape());
    }
}

class square implements shape {
    public void transform() {
        System.out.println("square transforms into "
+ shapeshifter.getrandomshape());
    }
}

class triangle implements shape {
    public void transform() {
        System.out.println("triangle transforms into
" + shapeshifter.getrandomshape());
    }
}

class shapeshifter {
    static String[] shapes = {"circle", "square",
"triangle"};
```

```
    static String getrandomshape() {
        return shapes[new
Random().nextInt(shapes.length)];
    }

    static void process(shape s) {
        s.transform();
    }

    public static void main(String[] args) {
        shape[] shapelist = {new circle(), new
square(), new triangle()};
        for(int i = 0; i < 5; i++) {
            process(shapelist[new
Random().nextInt(shapelist.length)]);
        }
    }
}
```

Output:



```
PS C:\Users\savit>  & 'C:\Program
a-project\bin' 'shapeshifter'
square transforms into square
circle transforms into square
triangle transforms into triangle
square transforms into circle
circle transforms into square
PS C:\Users\savit> 
```

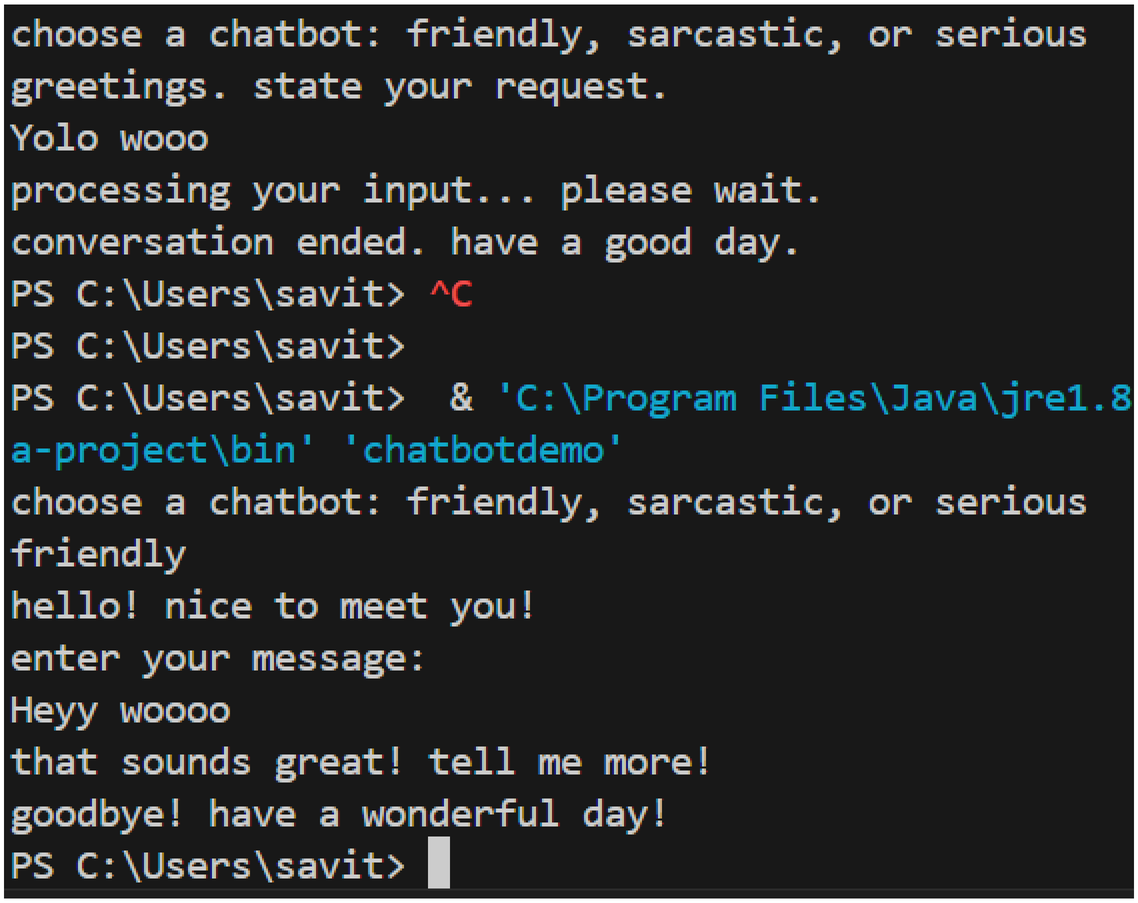
(b) AI Chatbots using interface commands:

Aim: Uses **interfaces and polymorphism** to create chatbot personalities. The chatbot interface ensures all chatbot types (friendlybot, sarcasticbot, seriousbot) have a respond method while allowing them to provide unique responses.

Program Code:

```java
import java.util.Scanner;

interface chatbot {
    String greet();
    String respond(String message);
    String farewell();
}

class friendlybot implements chatbot {
    public String greet() {
        return "hello! nice to meet you!";
    }
    public String respond(String message) {
        return "that sounds great! tell me more!";
    }
    public String farewell() {
        return "goodbye! have a wonderful day!";
    }
}

class sarcasticbot implements chatbot {
    public String greet() {
        return "oh, wonderful, another
conversation...";
    }
    public String respond(String message) {
        return "wow, what a unique and totally
original thought!";
    }
    public String farewell() {
        return "finally, some peace and quiet. bye!";
    }
}

class seriousbot implements chatbot {
    public String greet() {
        return "greetings. state your request.";
    }
    public String respond(String message) {
        return "processing your input... please
wait.";
    }
    public String farewell() {
        return "conversation ended. have a good
day.";
    }
}

class chatbotdemo {
    static chatbot getchatbot(String choice) {
        if (choice.equals("friendly")) return new
friendlybot();
        if (choice.equals("sarcastic")) return new
sarcasticbot();
        if (choice.equals("serious")) return new
seriousbot();
        return null;
    }
    public static void main(String[] args) {
        Scanner scanner = new Scanner(System.in);
        System.out.println("choose a chatbot: 
friendly, sarcastic, or serious");
        String choice = 
scanner.nextLine().toLowerCase();

        chatbot bot = getchatbot(choice);
        if (bot == null) {
            System.out.println("invalid choice. 
exiting.");
            scanner.close();
            return;
        }

        System.out.println(bot.greet());
        System.out.println("enter your message:");
        String message = scanner.nextLine();
        System.out.println(bot.respond(message));
        System.out.println(bot.farewell());

        scanner.close();
    }
}
```

Output:



```
choose a chatbot: friendly, sarcastic, or serious
greetings. state your request.
Yolo wooo
processing your input... please wait.
conversation ended. have a good day.
PS C:\Users\savit> ^C
PS C:\Users\savit>
PS C:\Users\savit>  & 'C:\Program Files\Java\jre1.8
a-project\bin' 'chatbotdemo'
choose a chatbot: friendly, sarcastic, or serious
friendly
hello! nice to meet you!
enter your message:
Heyy woooo
that sounds great! tell me more!
goodbye! have a wonderful day!
PS C:\Users\savit>
```

(c) Controlling Music actions with methods defined different for each class using Interface calling.

Aim: This program uses interfaces to create a flexible way of handling **different musical instruments**. The ins interface provides common actions (play, tune, stop), and each instrument (pi, gu, dr) gives its own version of these actions. The user picks an instrument

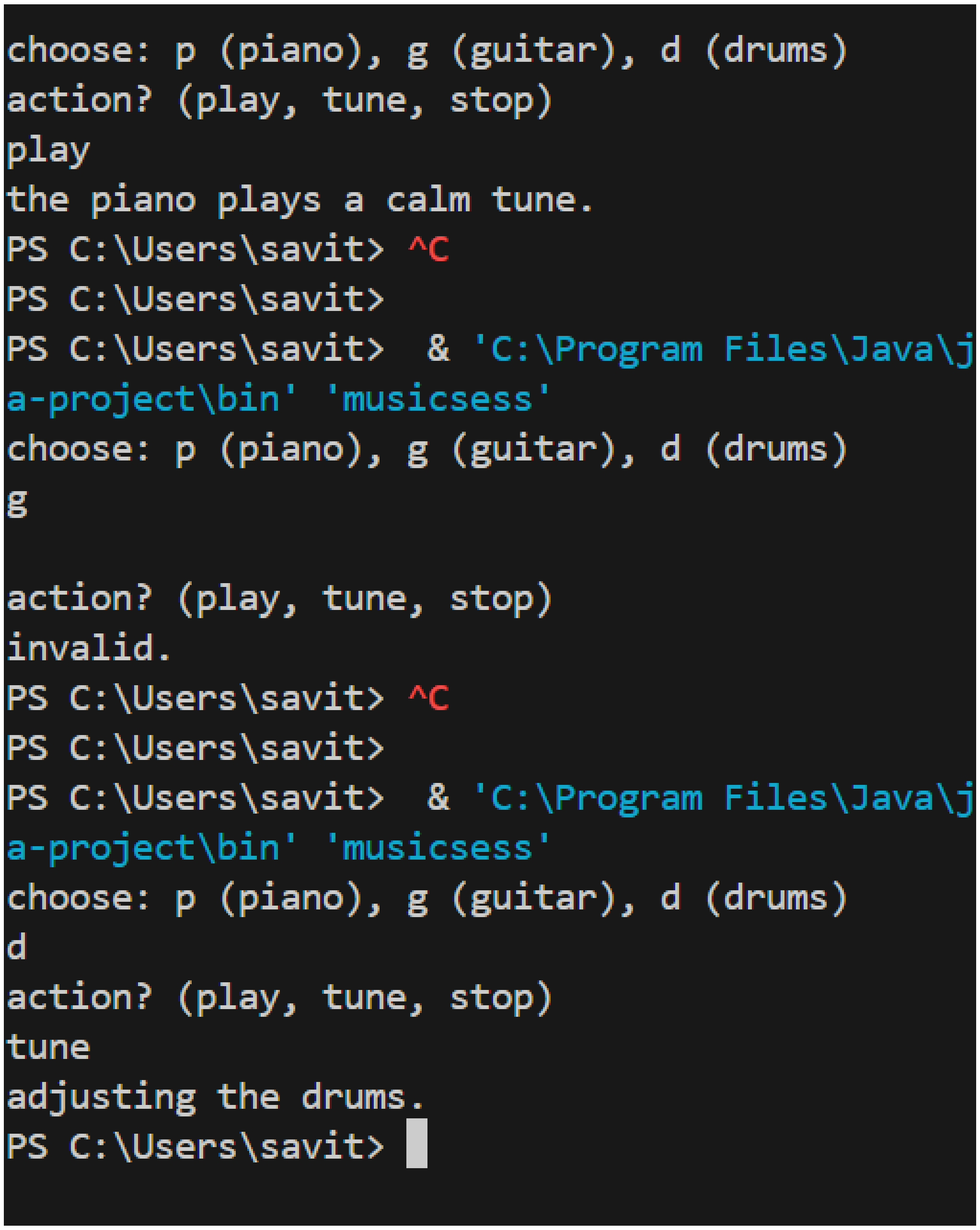
and decides what to do, showing how interfaces allow multiple classes to share a ***common structure*** while behaving differently.

Program Code:

```
import java.util.Scanner;

interface ins {
    void play();
    void tune();
    void stop();
}

class pi implements ins {
    public void play() {
        System.out.println("the piano plays a calm 
tune.");
    }
    public void tune() {
        System.out.println("tuning the piano.");
    }
    public void stop() {
        System.out.println("the piano stops.");
    }
}

class gu implements ins {
    public void play() {
        System.out.println("the guitar plays a nice 
rhythm.");
    }
    public void tune() {
        System.out.println("tightening the guitar.");
    }
    public void stop() {
        System.out.println("the guitar stops.");
    }
}

class dr implements ins {
    public void play() {
        System.out.println("the drums make a loud 
beat.");
    }
    public void tune() {
        System.out.println("adjusting the drums.");
    }
    public void stop() {
        System.out.println("the drums stop.");
    }
}

class musicsess {
    static ins getins(String choice) {
        if (choice.equals("p")) return new pi();
        if (choice.equals("g")) return new gu();
        if (choice.equals("d")) return new dr();
        return null;
    }

    public static void main(String[] args) {
        Scanner s = new Scanner(System.in);
        System.out.println("choose: p (piano), g 
(guitar), d (drums)");
        String choice = s.nextLine().toLowerCase();
```

```java
        ins selected = getins(choice);
        if (selected == null) {
            System.out.println("not an option.");
            s.close();
            return;
        }

        System.out.println("action? (play, tune,
stop)");
        String action = s.nextLine().toLowerCase();

        switch(action) {
            case "play": selected.play(); break;
            case "tune": selected.tune(); break;
            case "stop": selected.stop(); break;
            default: System.out.println("invalid.");
        }

        s.close();
    }
}
```

Output:



```
choose: p (piano), g (guitar), d (drums)
action? (play, tune, stop)
play
the piano plays a calm tune.
PS C:\Users\savit> ^C
PS C:\Users\savit>
PS C:\Users\savit>  & 'C:\Program Files\Java\j
a-project\bin' 'musicsess'
choose: p (piano), g (guitar), d (drums)
g

action? (play, tune, stop)
invalid.
PS C:\Users\savit> ^C
PS C:\Users\savit>
PS C:\Users\savit>  & 'C:\Program Files\Java\j
a-project\bin' 'musicsess'
choose: p (piano), g (guitar), d (drums)
d
action? (play, tune, stop)
tune
adjusting the drums.
PS C:\Users\savit>
```

(d) Simple Game Simulator:

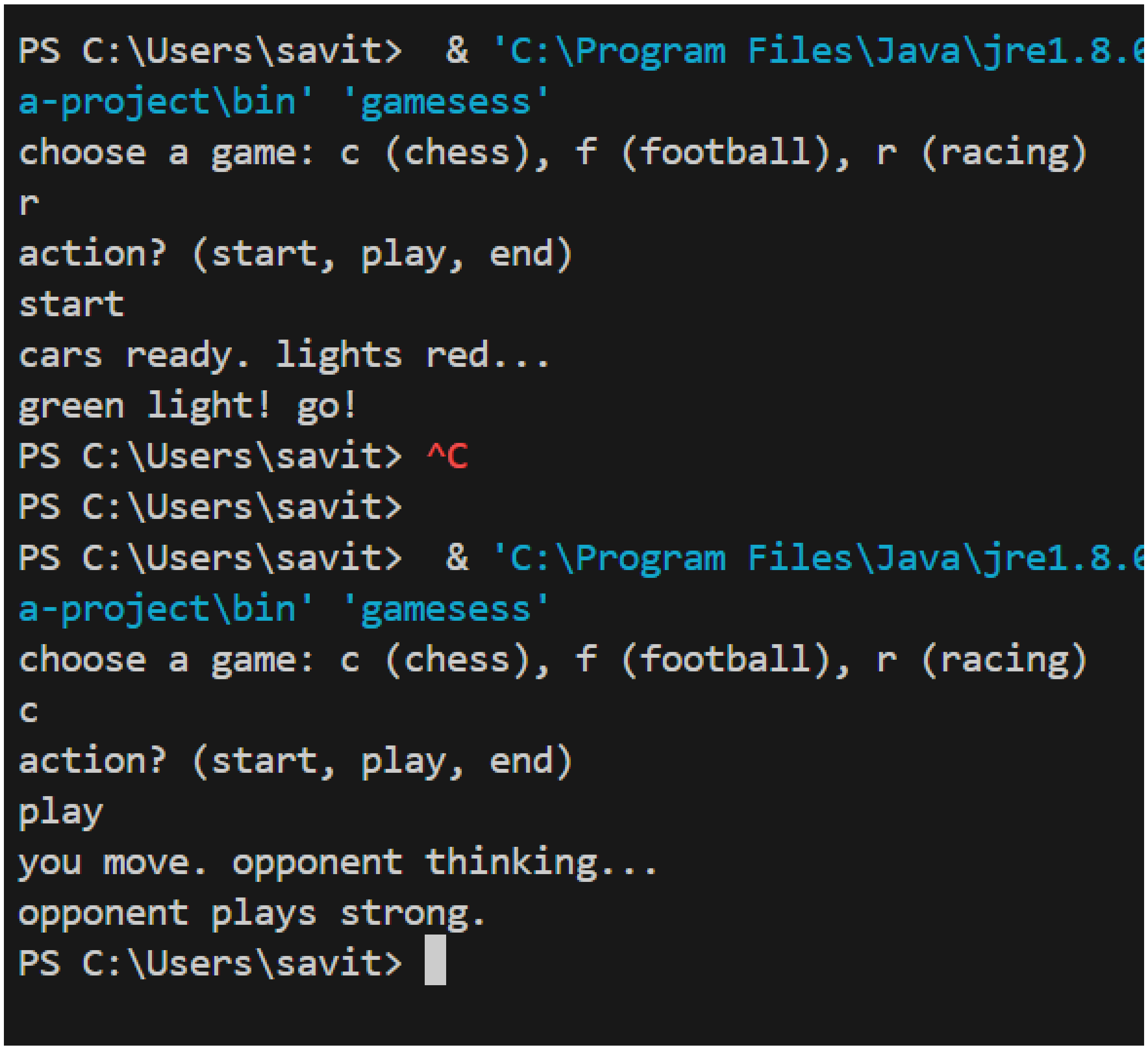
Aim: we use interface to create a flexible game system. The game interface provides common actions (start, play, end), and each game (chess, football, racing) has its own version of these actions. The user picks a game and interacts with it.

Program Code:

```
import java.util.Scanner;
import java.util.Random;

interface game {
    void start();
    void play();
    void end();
}

class chess implements game {
    Random rand = new Random();
    public void start() {
        System.out.println("chess starts. white moves
first.");
    }
    public void play() {
        System.out.println("you move. opponent
thinking...");
        System.out.println(rand.nextBoolean() ?
"opponent blunders!" : "opponent plays strong.");
    }
    public void end() {
        System.out.println(rand.nextBoolean() ? "you
win! checkmate!" : "you lose! try again.");
    }
}

class football implements game {
    Random rand = new Random();
    public void start() {
        System.out.println("match begins. kickoff!");
    }
    public void play() {
        System.out.println("shoot or pass? (s/p)");
        Scanner s = new Scanner(System.in);
        String choice = s.nextLine().toLowerCase();
        System.out.println(choice.equals("s") ?
(rand.nextBoolean() ? "goal!!!" : "missed!") : "you
passed the ball.");
    }
    public void end() {
        System.out.println(rand.nextBoolean() ? "your
team wins!" : "your team loses.");
    }
}

class racing implements game {
    Random rand = new Random();
    public void start() {
        System.out.println("cars ready. lights
red...");
        System.out.println("green light! go!");
    }
    public void play() {
```

```java
        System.out.println("speed up or slow down? (u/d)");
        Scanner s = new Scanner(System.in);
        String choice = s.nextLine().toLowerCase();
        System.out.println(choice.equals("u") ? "you speed up!" : "you slow down.");
    }
    public void end() {
        System.out.println(rand.nextBoolean() ? "you win the race!" : "you lose. try again.");
    }
}

class gamesess {
    static game getgame(String choice) {
        if (choice.equals("c")) return new chess();
        if (choice.equals("f")) return new football();
        if (choice.equals("r")) return new racing();
        return null;
    }

    public static void main(String[] args) {
        Scanner s = new Scanner(System.in);
        System.out.println("choose a game: c (chess), f (football), r (racing)");
        String choice = s.nextLine().toLowerCase();

        game selected = getgame(choice);
        if (selected == null) {
            System.out.println("not an option.");
            s.close();
            return;
        }
        System.out.println("action? (start, play,
end)");
        String action = s.nextLine().toLowerCase();

        switch(action) {
            case "start": selected.start(); break;
            case "play": selected.play(); break;
            case "end": selected.end(); break;
            default: System.out.println("invalid.");
        }

        s.close();
    }
}
```

Output:



```
PS C:\Users\savit>  & 'C:\Program Files\Java\jre1.8.6
a-project\bin' 'gamesess'
choose a game: c (chess), f (football), r (racing)
r
action? (start, play, end)
start
cars ready. lights red...
green light! go!
PS C:\Users\savit> ^C
PS C:\Users\savit>
PS C:\Users\savit>  & 'C:\Program Files\Java\jre1.8.6
a-project\bin' 'gamesess'
choose a game: c (chess), f (football), r (racing)
c
action? (start, play, end)
play
you move. opponent thinking...
opponent plays strong.
PS C:\Users\savit>
```

Abstraction:

13. Abstraction Programs:

(a) **Weather Forecast** Display:

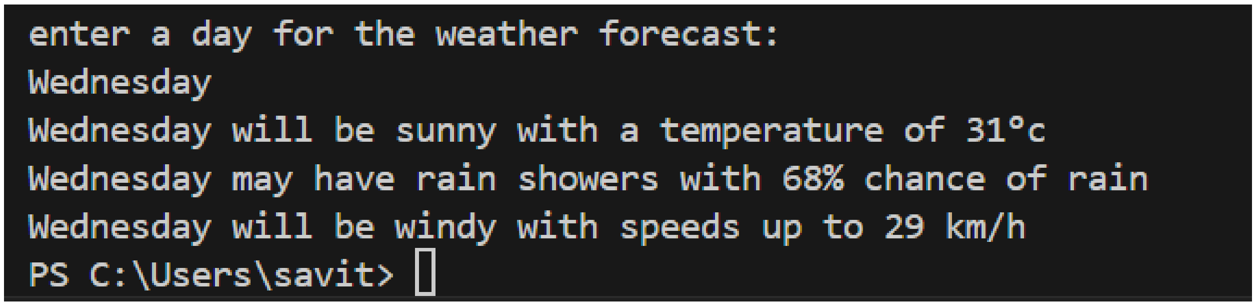
Aim: To demonstrate how different weather predictions can be handled using interface-based abstraction in Java.

Program Code:

```java
import java.util.Random;
import java.util.Scanner;

abstract class forecaster {
    abstract String predict(String day);
}

class sunnyforecaster extends forecaster {
    public String predict(String day) {
        int temp = new Random().nextInt(10) + 25;
        return day + " will be sunny with a
temperature of " + temp + "°c";
    }
}

class rainyforecaster extends forecaster {
    public String predict(String day) {
        int rainchance = new Random().nextInt(40) +
60;
        return day + " may have rain showers with " +
rainchance + "% chance of rain";
    }
}

class windyforecaster extends forecaster {
    public String predict(String day) {
        int windspeed = new Random().nextInt(20) +
10;
        return day + " will be windy with speeds up
to " + windspeed + " km/h";
    }
}

public class weatherstation {
    static void forecast(forecaster f, String day) {
        System.out.println(f.predict(day));
    }

    public static void main(String[] args) {
        Scanner scanner = new Scanner(System.in);
        System.out.println("enter a day for the
weather forecast:");
        String day = scanner.nextLine();
        scanner.close();

        forecaster[] forecasts = {new
sunnyforecaster(), new rainyforecaster(), new
windyforecaster()};
        for (forecaster f : forecasts) forecast(f,
day);
    }
```

```
}
```

Output:



```
enter a day for the weather forecast:
Wednesday
Wednesday will be sunny with a temperature of 31°c
Wednesday may have rain showers with 68% chance of rain
Wednesday will be windy with speeds up to 29 km/h
PS C:\Users\savit>
```

(b) Jukebox Music Player:

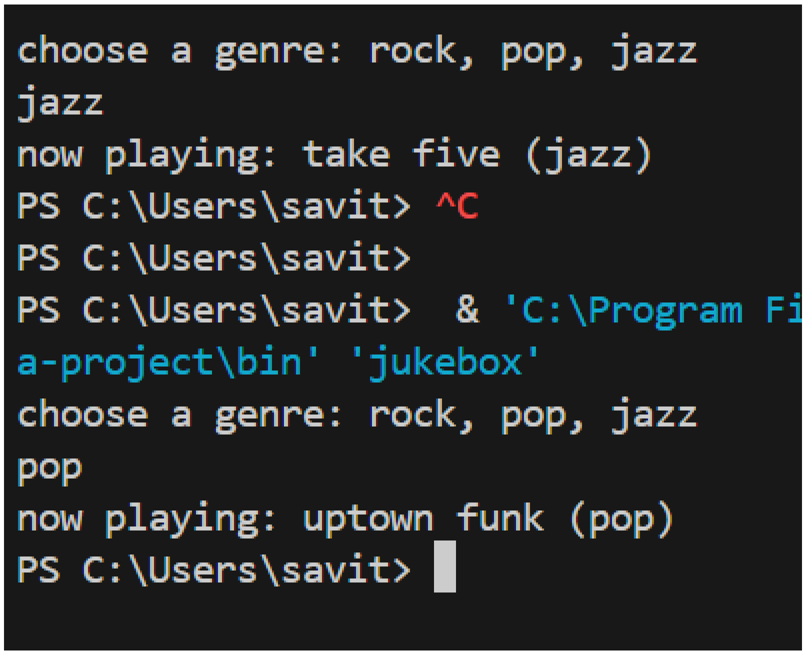
Aim: To create an **abstract music player system** that plays random songs from different genres (rock, pop, and jazz) based on user selection.

Program Code:

```
import java.util.Random;
import java.util.Scanner;

abstract class musicplayer {
    abstract String play();
}

class rockplayer extends musicplayer {
    public String play() {
        String[] songs = {"thunderstruck",
"bohemian rhapsody", "sweet child o' mine"};
        return "now playing: " + songs[new
Random().nextInt(songs.length)] + " (rock)";
    }
}

class popplayer extends musicplayer {
    public String play() {
        String[] songs = {"blinding lights",
"shake it off", "uptown funk"};
        return "now playing: " + songs[new
Random().nextInt(songs.length)] + " (pop)";
    }
}

class jazzplayer extends musicplayer {
    public String play() {
        String[] songs = {"take five", "so
what", "fever"};
        return "now playing: " + songs[new
Random().nextInt(songs.length)] + " (jazz)";
    }
}

public class jukebox {
    public static void main(String[] args) {
        Scanner scanner = new
Scanner(System.in);
```

```
        System.out.println("choose a genre: 
rock, pop, jazz");
        String choice = 
scanner.nextLine().toLowerCase();
        scanner.close();

        musicplayer player;
        if (choice.equals("rock")) player = new 
rockplayer();
        else if (choice.equals("pop")) player = 
new popplayer();
        else if (choice.equals("jazz")) player 
= new jazzplayer();
        else {
            System.out.println("that is invalid 
genre.");
            return;
        }

        System.out.println(player.play());
    }
}
```

Output:



```
choose a genre: rock, pop, jazz
jazz
now playing: take five (jazz)
PS C:\Users\savit> ^C
PS C:\Users\savit>
PS C:\Users\savit>  & 'C:\Program Fi
a-project\bin' 'jukebox'
choose a genre: rock, pop, jazz
pop
now playing: uptown funk (pop)
PS C:\Users\savit>
```

(c) Movie Recommendation Machine:

Aim: To provide **movie suggestions based on user-selected genre** (action, comedy, or sci-fi) using abstraction.

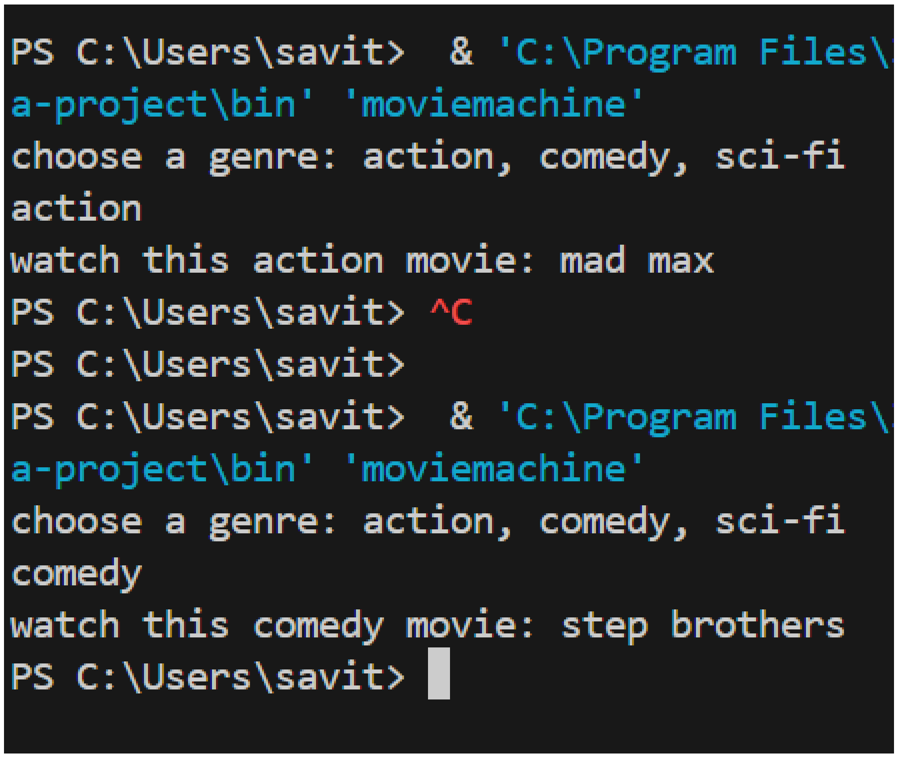
Program Code:

```
import java.util.Random;
import java.util.Scanner;

abstract class recommender {
    abstract String suggest();
}
```

```
class actionrecommender extends recommender {
    public String suggest() {
        String[] movies = {"mad max", "john wick",
"gladiator"};
        return "watch this action movie: " + movies[new
Random().nextInt(movies.length)];
    }
}

class comedymovierecommender extends recommender {
    public String suggest() {
        String[] movies = {"the mask", "superbad", "step
brothers"};
        return "watch this comedy movie: " + movies[new
Random().nextInt(movies.length)];
    }
}

class scifirecommender extends recommender {
    public String suggest() {
        String[] movies = {"inception", "interstellar", "the
matrix"};
        return "watch this sci-fi movie: " + movies[new
Random().nextInt(movies.length)];
    }
}

public class moviemachine {
    public static void main(String[] args) {
        Scanner scanner = new Scanner(System.in);
        System.out.println("choose a genre: action, comedy,
sci-fi");
        String genre = scanner.nextLine().toLowerCase();
        scanner.close();

        recommender rec;
        if (genre.equals("action")) rec = new
actionrecommender();
```

```
        else if (genre.equals("comedy")) rec = new
comedymovierecommender();
        else if (genre.equals("sci-fi")) rec = new
scifirecommender();
        else {
            System.out.println("unknown genre");
            return;
        }

        System.out.println(rec.suggest());
    }
}
```

Output:



```
PS C:\Users\savit>  & 'C:\Program Files\
a-project\bin' 'moviemachine'
choose a genre: action, comedy, sci-fi
action
watch this action movie: mad max
PS C:\Users\savit> ^C
PS C:\Users\savit>
PS C:\Users\savit>  & 'C:\Program Files\
a-project\bin' 'moviemachine'
choose a genre: action, comedy, sci-fi
comedy
watch this comedy movie: step brothers
PS C:\Users\savit>
```

(d) Meal recommendation advisor

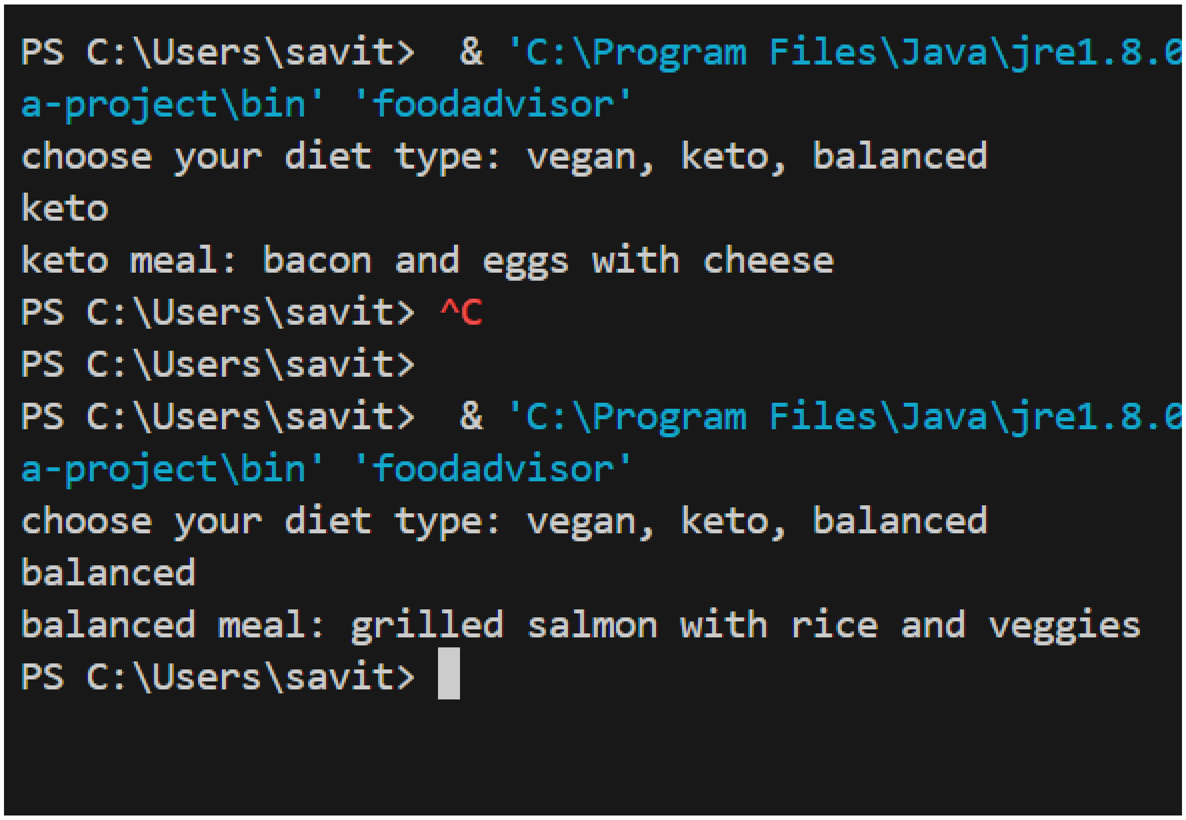
Aim: To create **a java program that suggests a meal option** based on the user's ***chosen dietary preference*** such as ***vegan, keto, or balanced***, by selecting a random meal from predefined options

Program Code:

```java
import java.util.Random;
import java.util.Scanner;

abstract class mealplanner {
    abstract String recommend();
}

class veganplanner extends mealplanner {
    public String recommend() {
        String[] meals = {
            "quinoa salad with chickpeas",
            "tofu stir fry with brown rice",
            "lentil soup and whole grain bread"
        };
        return "vegan meal: " + meals[new
Random().nextInt(meals.length)];
    }
}

class ketoplanner extends mealplanner {
    public String recommend() {
        String[] meals = {
            "grilled chicken with avocado",
            "bacon and eggs with cheese",
            "zucchini noodles with pesto"
        };
        return "keto meal: " + meals[new
Random().nextInt(meals.length)];
    }
}

class balancedplanner extends mealplanner {
    public String recommend() {
        String[] meals = {
            "grilled salmon with rice and veggies",
            "chicken wrap with hummus and salad",
            "spaghetti with tomato sauce and fruit salad"
        };
        return "balanced meal: " + meals[new
Random().nextInt(meals.length)];
    }
}

public class foodadvisor {
    public static void main(String[] args) {
        Scanner scanner = new Scanner(System.in);
        System.out.println("choose your diet type: vegan,
keto, balanced");
        String diet = scanner.nextLine().toLowerCase();
        scanner.close();

        mealplanner planner;
        if (diet.equals("vegan")) planner = new
veganplanner();
        else if (diet.equals("keto")) planner = new
ketoplanner();
        else if (diet.equals("balanced")) planner = new
balancedplanner();
        else {
            System.out.println("unknown diet type");
            return;
        }

        System.out.println(planner.recommend());
    }
```

```
}
```

Output:



```
PS C:\Users\savit>  & 'C:\Program Files\Java\jre1.8.0
a-project\bin' 'foodadvisor'
choose your diet type: vegan, keto, balanced
keto
keto meal: bacon and eggs with cheese
PS C:\Users\savit> ^C
PS C:\Users\savit>
PS C:\Users\savit>  & 'C:\Program Files\Java\jre1.8.0
a-project\bin' 'foodadvisor'
choose your diet type: vegan, keto, balanced
balanced
balanced meal: grilled salmon with rice and veggies
PS C:\Users\savit>
```

Encapsulation

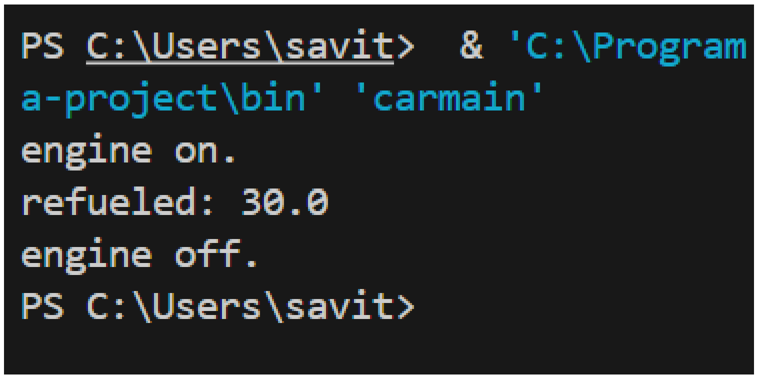
14**(a)** Car maintenance system using Interface:

Aim: this program uses an interface to control a car. it lets you start, stop, and refuel the car, showing how interfaces help in organizing common actions in different classes.

Program Code:

```java
class car {
    private boolean on;
    private double fuellvl;

    public car(double fuellvl) {
        this.on = false;
        this.fuellvl = fuellvl;
    }

    public boolean ison() {
        return on;
    }

    public double getfuel() {
        return fuellvl;
    }
    public void start() {
        if (fuellvl > 0) {
            on = true;
            System.out.println("engine on.");
        } else {
            System.out.println("no fuel.");
        }
    }

    public void stop() {
        on = false;
        System.out.println("engine off.");
    }

    public void refuel(double amt) {
        fuellvl += amt;
        System.out.println("refueled: " + fuellvl);
    }
}

public class carmain {
    public static void main(String[] args) {
        car mycar = new car(10.0);

        mycar.start();
        mycar.refuel(20.0);
        mycar.stop();
    }
}
```

Output:



```
PS C:\Users\savit>  & 'C:\Program
a-project\bin' 'carmain'
engine on.
refueled: 30.0
engine off.
PS C:\Users\savit>
```

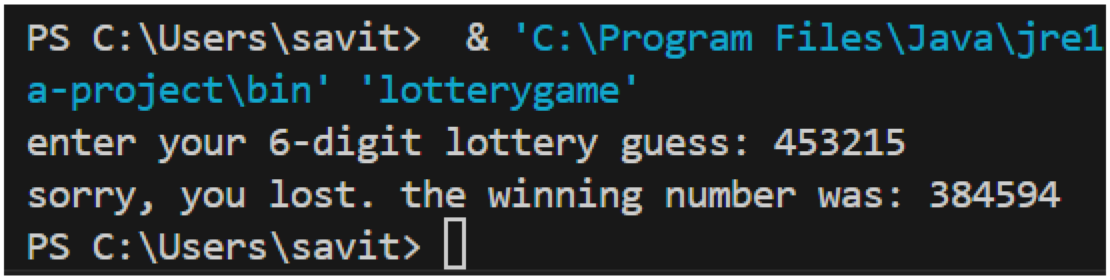
(b) Lottery System with encapsulating lottery number:

Aim: To create a lottery system where the winning number is a random 6 – digit code, encapsulating it; and the user enters a six digit code to guess the lottery number.

Program Code:

```java
import java.util.Scanner;

class lottery {
    private int lotterynum;

    public lottery(int lotteryval) {
        this.lotterynum = lotteryval;
    }

    public int getlottery() {
        return lotterynum;
    }
    public void verifylottery(int guessednum) {
        if (guessednum == lotterynum) {
            System.out.println("congrats! you won 1 million 
dollars!");
        } else {
            System.out.println("sorry, you lost. the winning 
number was: " + getlottery());
        }
    }
}

public class lotterygame {
    public static void main(String[] args) {
        Scanner scanner = new Scanner(System.in);
        int lotteryvalue = (int) (Math.random() * 900000) + 
100000;
        lottery lottery = new lottery(lotteryvalue);
        System.out.print("enter your 6-digit lottery guess: 
");
        int userguess = scanner.nextInt();
        lottery.verifylottery(userguess);
        scanner.close();
    }
}
```

Output:



```
PS C:\Users\savit>  & 'C:\Program Files\Java\jre1
a-project\bin' 'lotterygame'
enter your 6-digit lottery guess: 453215
sorry, you lost. the winning number was: 384594
PS C:\Users\savit> 
```

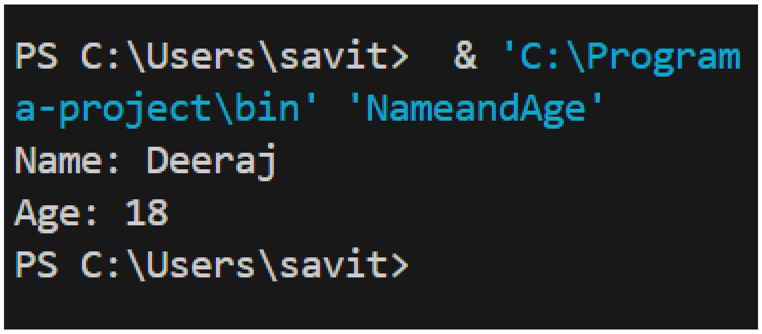
(c) Name and Age:

Aim: To receive Name and Age from User and encapsulate it, and when requested, show the user’s name and age.

Program Code:

```java
class Person {
    private String name;
    private int age;

    public String getname() {
        return name;
    }

    public void setname(String name) {
        this.name = name;
    }

    public int getage() {
        return age;
    }

    public void setage(int age) {
        this.age = age;
    }
}

public class NameandAge {
    public static void main(String[] args) {
        Person p = new Person();
        p.setname("Deeraj");
        p.setage(18);
        System.out.println("Name: " + p.getname());
        System.out.println("Age: " + p.getage());
    }
}
```

Output:



```
PS C:\Users\savit>  & 'C:\Program
a-project\bin' 'NameandAge'
Name: Deeraj
Age: 18
PS C:\Users\savit>
```

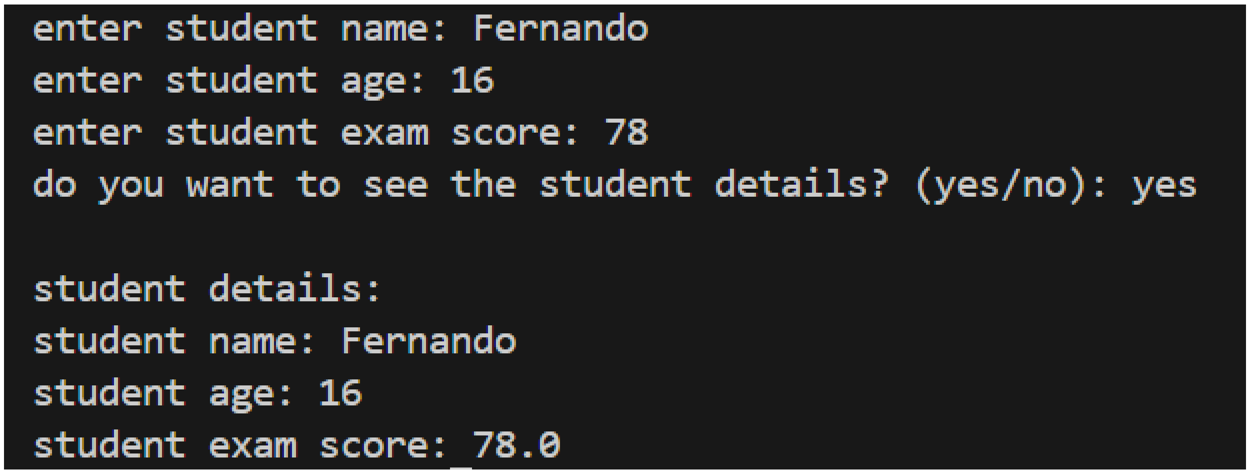
(d) School student information and grade encapsulation:

Aim: To make student information private and display the information (test grades) only through getter method.

Program Code:

```
import java.util.Scanner;

class student {
    private String name;
    private int age;
    private double grade;

    public student(String name, int age, double grade) {
        this.name = name;
        this.age = age;
        this.grade = grade;
    }

    public String getname() {
        return name;
    }

    public int getage() {
        return age;
    }

    public double getgrade() {
        return grade;
    }

    public void studentinfo() {
        System.out.println("student name: " + getname());
        System.out.println("student age: " + getage());
        System.out.println("student exam score: " +
getgrade());
    }
}

public class studentapp {
    public static void main(String[] args) {
        Scanner scanner = new Scanner(System.in);

        System.out.print("enter student name: ");
        String name = scanner.nextLine();

        System.out.print("enter student age: ");
        int age = scanner.nextInt();

        System.out.print("enter student exam score: ");
        double grade = scanner.nextDouble();
        scanner.nextLine();

        student stu = new student(name, age, grade);
        System.out.print("do you want to see the student details? (yes/no): ");
        String choice = scanner.nextLine().toLowerCase();

        if (choice.equals("yes")) {
            System.out.println("\nstudent details:");
            stu.studentinfo();
        } else {
            System.out.println("okay, details not shown.");
        }

        scanner.close();
    }
}
```

Output:



```
enter student name: Fernando
enter student age: 16
enter student exam score: 78
do you want to see the student details? (yes/no): yes

student details:
student name: Fernando
student age: 16
student exam score: 78.0
```

Packages

15. Packages Programs in Java:

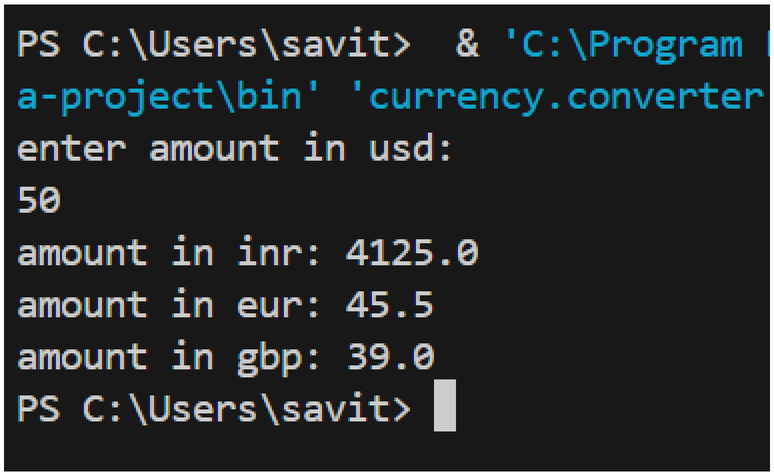
(a) Currency conversion:

Aim: To convert one form of currency to another using **user defined packages in java**

Program Code:

```java
package currency;
import java.util.Scanner;

public class converter {
    public static void main(String[] args) {
        Scanner scanner = new Scanner(System.in);
        System.out.println("enter amount in usd:");
        double usd = scanner.nextDouble();
        double inr = usd * 82.50;
        double eur = usd * 0.91;
        double gbp = usd * 0.78;
        System.out.println("amount in inr: " + inr);
        System.out.println("amount in eur: " + eur);
        System.out.println("amount in gbp: " + gbp);
        scanner.close();
    }
}
```

Output:



```
PS C:\Users\savit>  & 'C:\Program
a-project\bin' 'currency.converter
enter amount in usd:
50
amount in inr: 4125.0
amount in eur: 45.5
amount in gbp: 39.0
PS C:\Users\savit>
```

(b) Random Joke Provider:

Aim: To print out randomly selected jokes by creating a dedicated package in Java.

Program Code:

```
package jokes;
import java.util.Random;

public class jokeprovider {
    public static void main(String[] args) {
        String[] jokes = {
            "why did the computer catch a cold? because it
left its windows open!",
            "why don’t programmers like nature? it has too
many bugs.",
            "how do you comfort a javascript bug? you 
console it.",
            "Why did the Java student bring a broom to the 
lab? Because they wanted to clean up their code and fix all 
the bugs!"
        };
        Random random = new Random();
        System.out.println(jokes[random.nextInt(jokes.length
)]);
    }
}
```

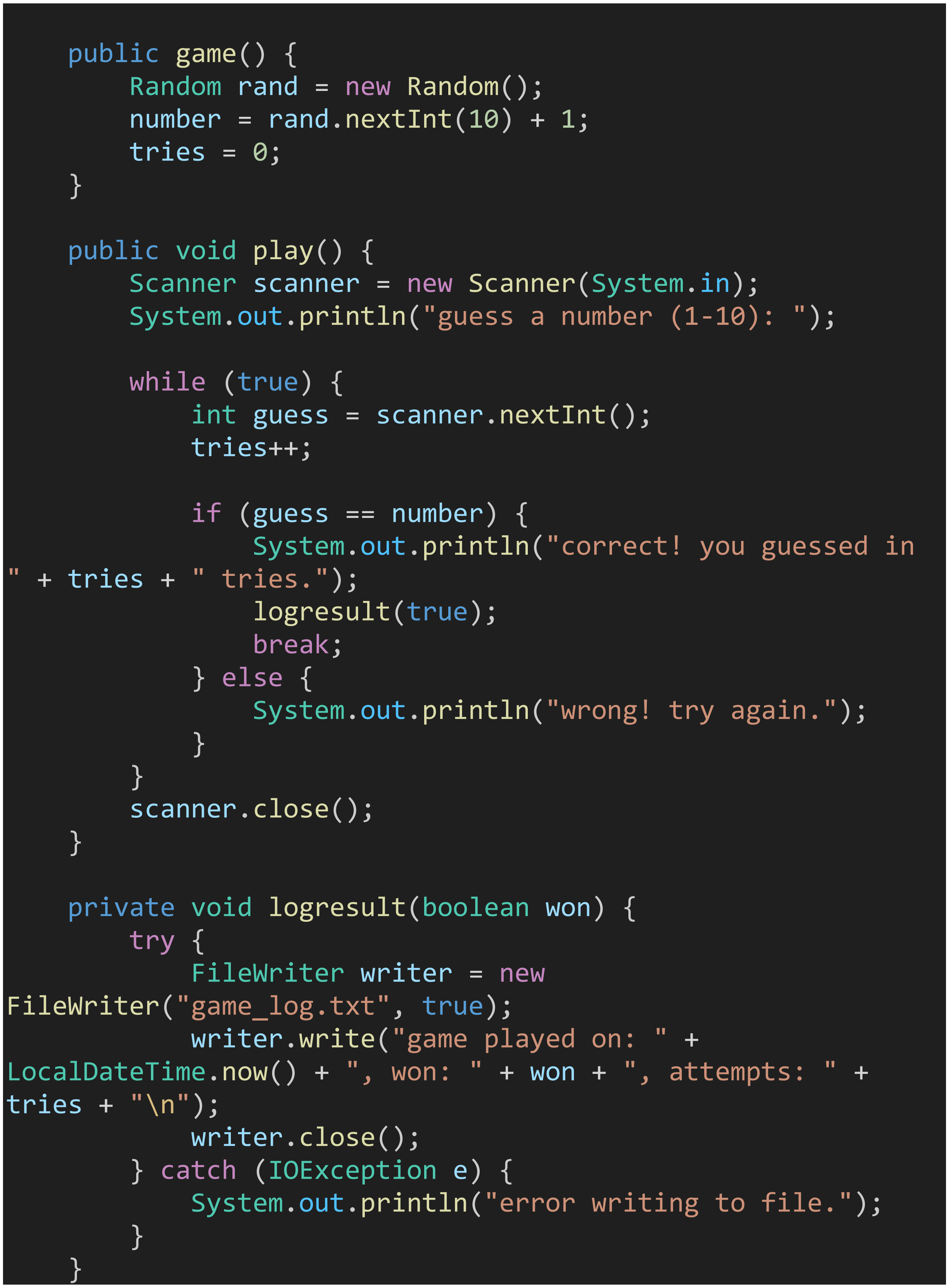
Built- In package:

(c) Random Number Guesser:

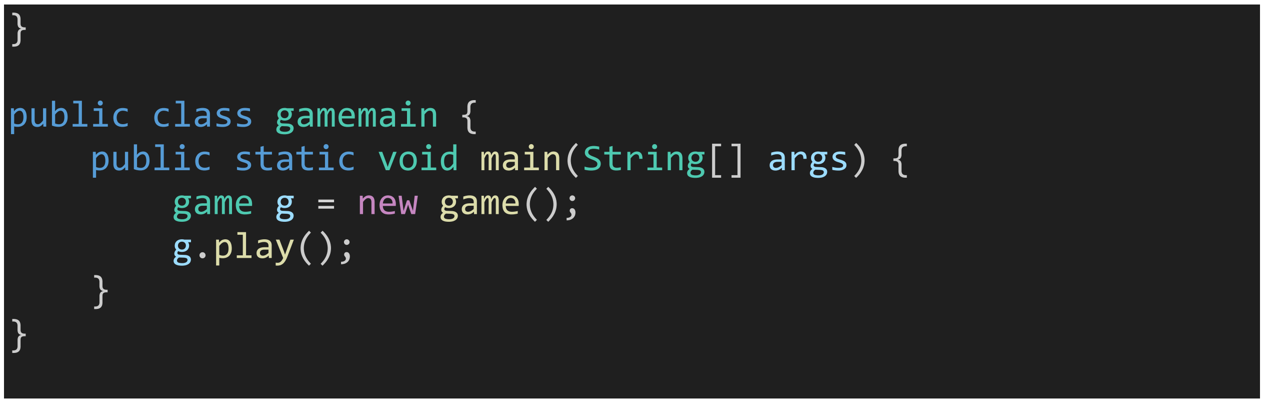
Aim: To guess a number (1-10) and the number of tries are stored in a .txt file. Once the number guessed is correct the number of tries are shown.

Program Code:

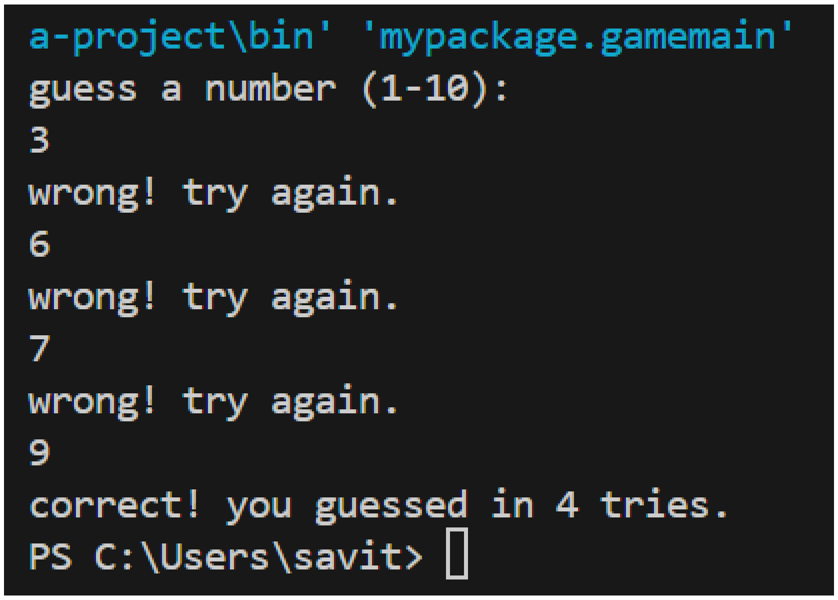
```
package mypackage;

import java.util.Scanner;
import java.util.Random;
import java.io.FileWriter;
import java.io.IOException;
import java.time.LocalDateTime;

class game {
    private int number;
    private int tries;
```



```java
    public game() {
        Random rand = new Random();
        number = rand.nextInt(10) + 1;
        tries = 0;
    }

    public void play() {
        Scanner scanner = new Scanner(System.in);
        System.out.println("guess a number (1-10): ");

        while (true) {
            int guess = scanner.nextInt();
            tries++;

            if (guess == number) {
                System.out.println("correct! you guessed in
" + tries + " tries.");
                logresult(true);
                break;
            } else {
                System.out.println("wrong! try again.");
            }
        }
        scanner.close();
    }

    private void logresult(boolean won) {
        try {
            FileWriter writer = new
FileWriter("game_log.txt", true);
            writer.write("game played on: " +
LocalDateTime.now() + ", won: " + won + ", attempts: " +
tries + "\n");
            writer.close();
        } catch (IOException e) {
            System.out.println("error writing to file.");
        }
    }
```


```
}

public class gamemain {
    public static void main(String[] args) {
        game g = new game();
        g.play();
    }
}
```

Output:


```
a-project\bin' 'mypackage.gamemain'
guess a number (1-10):
3
wrong! try again.
6
wrong! try again.
7
wrong! try again.
9
correct! you guessed in 4 tries.
PS C:\Users\savit> 
```

(d) Journal Application and text file storing:

Aim: To receive journal entries an store them in a txt file, and display the contents of the file.

Program Code:

```
package journalapp;

import java.util.Scanner;
import java.io.FileWriter;
import java.io.FileReader;
import java.io.BufferedReader;
import java.io.IOException;
import java.time.LocalDateTime;

class diary {
    private String entry;
    private String timestamp;

    public void writeentry() {
        Scanner scanner = new Scanner(System.in);
        System.out.print("write your diary entry: ");
        entry = scanner.nextLine();
        timestamp = LocalDateTime.now().toString();
        saveentry();
        System.out.println("entry saved on: " + timestamp);
    }

    private void saveentry() {
        try (FileWriter writer = new FileWriter("diary.txt",
true)) {
            writer.write("entry on " + timestamp + ": " +
entry + "\n");
        } catch (IOException e) {
            System.out.println("error saving entry.");
        }
    }

    public void showentries() {
        try (FileReader reader = new
FileReader("diary.txt"); BufferedReader br = new
BufferedReader(reader)) {
            String line;
```

```java
            System.out.println("\nyour journal:");
            while ((line = br.readLine()) != null) {
                System.out.println(line);
            }
        } catch (IOException e) {
            System.out.println("error reading the
journal.");
        }
    }
}

public class diaryapp {
    public static void main(String[] args) {
        Scanner scanner = new Scanner(System.in);
        diary d = new diary();

        d.writeentry();

        System.out.print("\ndo you want to see all your
entries? (yes/no): ");
        String choice = scanner.nextLine().toLowerCase();

        if (choice.equals("yes")) {
            d.showentries();
        } else {
            System.out.println("okay, journal not shown.");
        }

        scanner.close();
    }
}
```

Output:

```
Write your diary entry: There once was a boy.
```

entry saved on: 2025-04-03 17:04:37

do you want to see all your entries? (yes/no): yes

There once was a boy.

Exception Handling

16. Exception Handling Programs in Java:

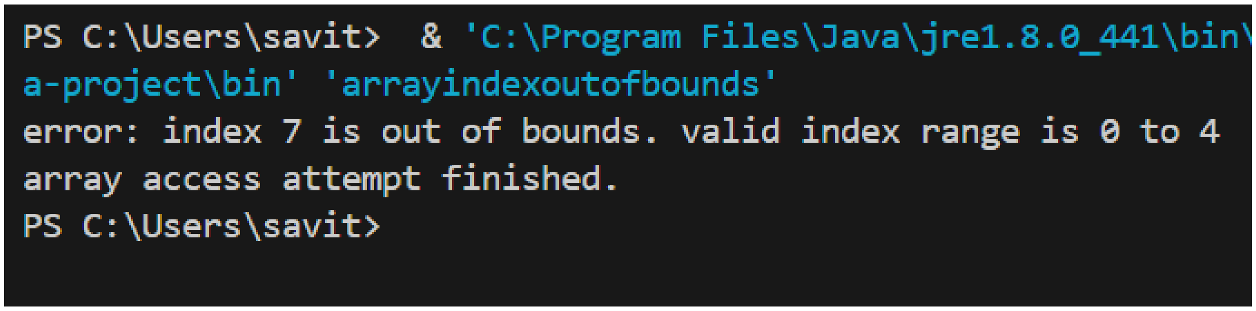
(a) Array index out of range error in java:

Aim: To display Array index out of bounds error in Java.

Program Code:

```
public class arrayindexoutofbounds {
    public static void main(String[] args) {
        int[] numbers = {1, 2, 3, 4, 5};
        try {
            System.out.println("trying to access index 7: "
+ numbers[7]);
        } catch (ArrayIndexOutOfBoundsException e) {
            System.out.println("error: index 7 is out of
bounds. valid index range is 0 to " + (numbers.length - 1));
        }
        System.out.println("array access attempt
finished.");
    }
}
```

Output:



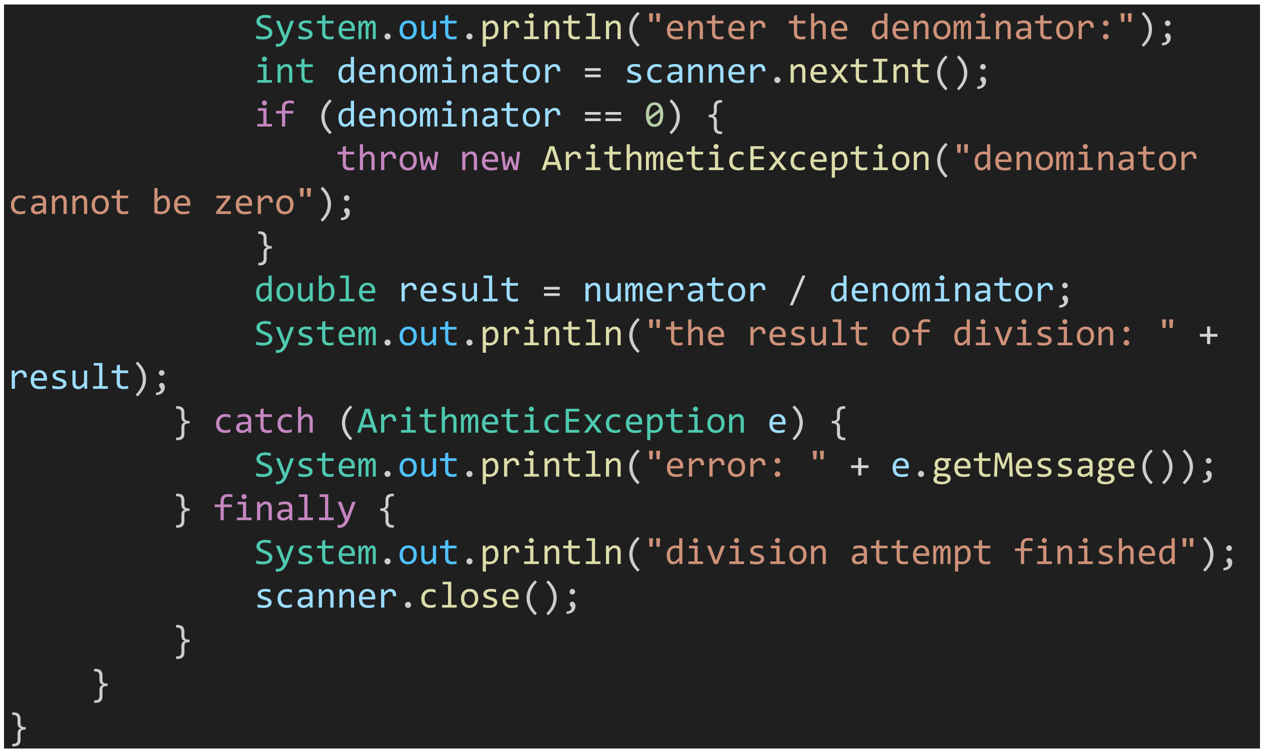
```
PS C:\Users\savit>  & 'C:\Program Files\Java\jre1.8.0_441\bin\
a-project\bin' 'arrayindexoutofbounds'
error: index 7 is out of bounds. valid index range is 0 to 4
array access attempt finished.
PS C:\Users\savit>
```

(b) Divide by Zero Error in Java:

Aim: to demonstrate error caused by division by zero in Java.

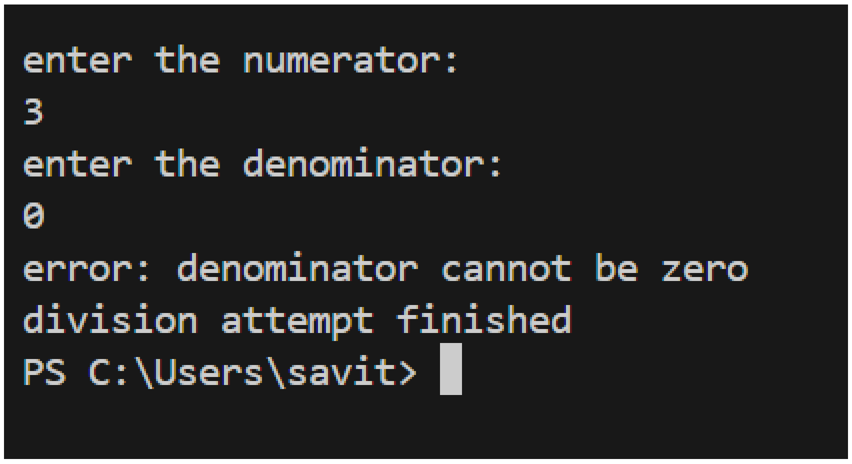
Program Code:

```
import java.util.Scanner;

public class dividebyzero {
    public static void main(String[] args) {
        Scanner scanner = new Scanner(System.in);
        try {
            System.out.println("enter the numerator:");
            int numerator = scanner.nextInt();
```



```
                System.out.println("enter the denominator:");
                int denominator = scanner.nextInt();
                if (denominator == 0) {
                    throw new ArithmeticException("denominator
cannot be zero");
                }
                double result = numerator / denominator;
                System.out.println("the result of division: " +
result);
            } catch (ArithmeticException e) {
                System.out.println("error: " + e.getMessage());
            } finally {
                System.out.println("division attempt finished");
                scanner.close();
            }
        }
}
```

Output:

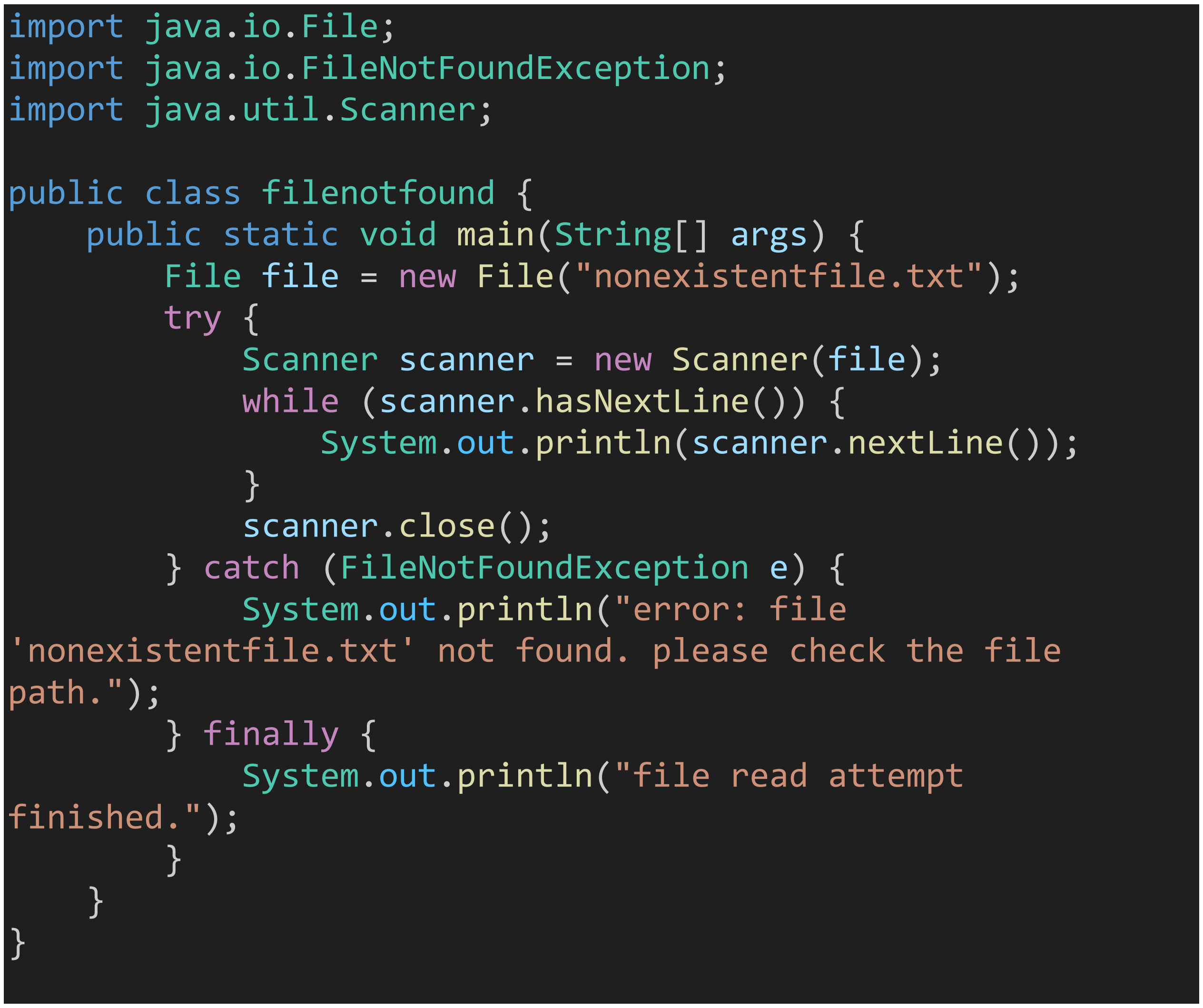


```
enter the numerator:
3
enter the denominator:
0
error: denominator cannot be zero
division attempt finished
PS C:\Users\savit>
```

(c) File Not Found Error in java:

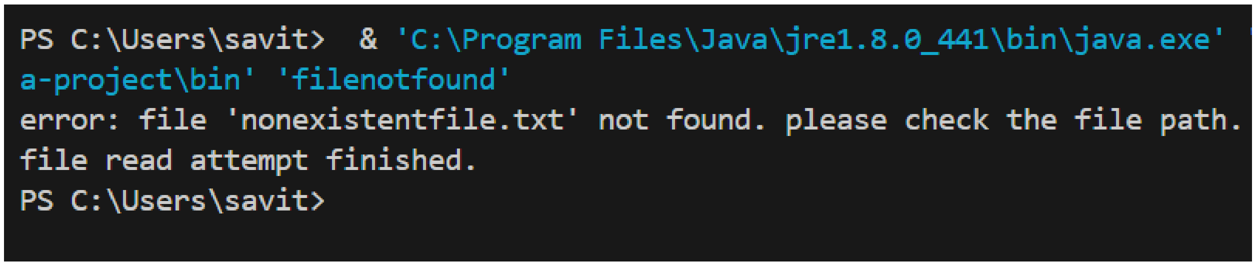
Aim: To show the error produced when the specified file does not exist if it is called.

Program Code:



```java
import java.io.File;
import java.io.FileNotFoundException;
import java.util.Scanner;

public class filenotfound {
    public static void main(String[] args) {
        File file = new File("nonexistentfile.txt");
        try {
            Scanner scanner = new Scanner(file);
            while (scanner.hasNextLine()) {
                System.out.println(scanner.nextLine());
            }
            scanner.close();
        } catch (FileNotFoundException e) {
            System.out.println("error: file
'nonexistentfile.txt' not found. please check the file
path.");
        } finally {
            System.out.println("file read attempt
finished.");
        }
    }
}
```

Output:



```
PS C:\Users\savit>  & 'C:\Program Files\Java\jre1.8.0_441\bin\java.exe'
a-project\bin' 'filenotfound'
error: file 'nonexistentfile.txt' not found. please check the file path.
file read attempt finished.
PS C:\Users\savit>
```

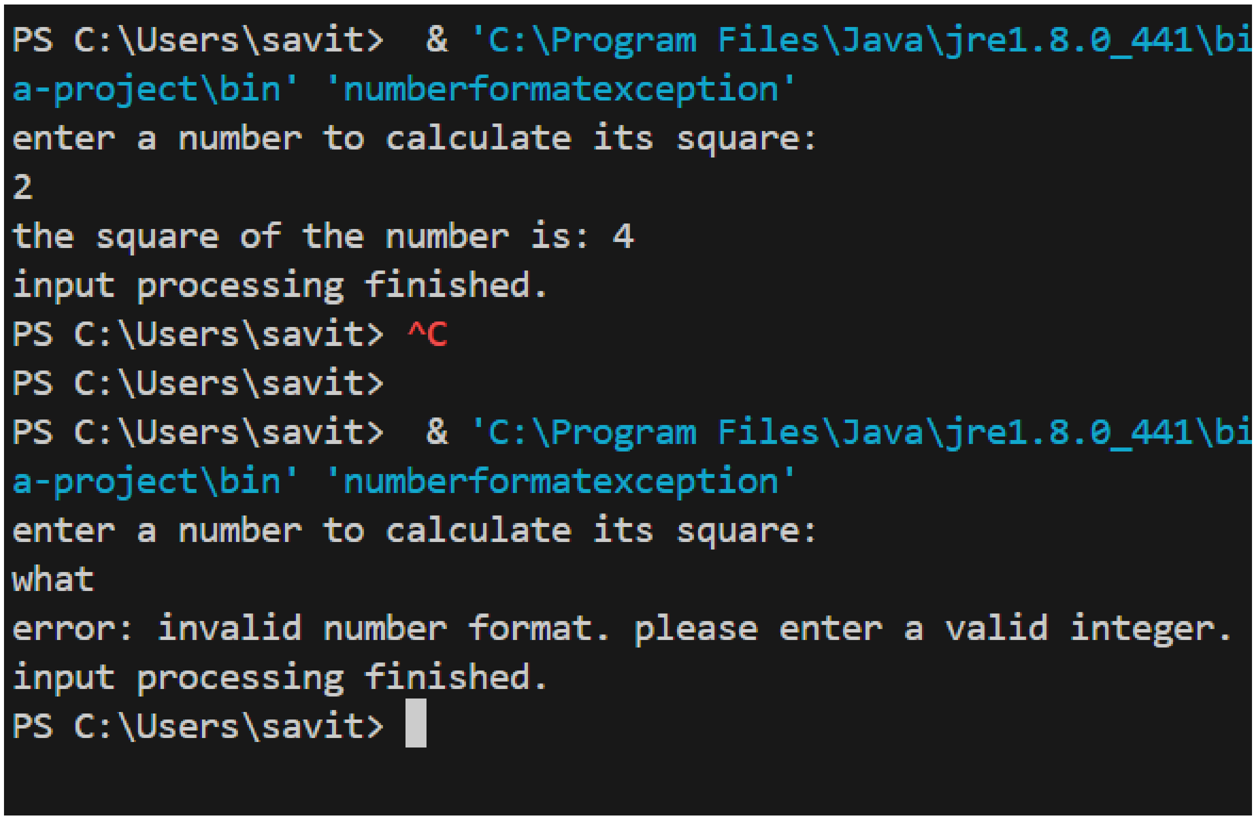
(d) Number Format Exception Error in Java:

Aim: this code handles **numberformatexception** by trying to convert **user input into an integer.** if the input is not a valid number, it catches the error and asks the user to enter a correct value.

Program Code:

```
import java.util.Scanner;

public class numberformatexception {
    public static void main(String[] args) {
        Scanner scanner = new Scanner(System.in);
        System.out.println("enter a number to
calculate its square:");
        try {
            String userInput = scanner.nextLine();
            int num = Integer.parseInt(userInput);
            System.out.println("the square of the
number is: " + (num * num));
        } catch (NumberFormatException e) {
            System.out.println("error: invalid number
format. please enter a valid integer.");
        } finally {
            System.out.println("input processing
finished.");
            scanner.close();
        }
    }
}
```

Output:



```
PS C:\Users\savit>  & 'C:\Program Files\Java\jre1.8.0_441\bi
a-project\bin' 'numberformatexception'
enter a number to calculate its square:
2
the square of the number is: 4
input processing finished.
PS C:\Users\savit> ^C
PS C:\Users\savit>
PS C:\Users\savit>  & 'C:\Program Files\Java\jre1.8.0_441\bi
a-project\bin' 'numberformatexception'
enter a number to calculate its square:
what
error: invalid number format. please enter a valid integer.
input processing finished.
PS C:\Users\savit>
```

File Handling

17. File Handling Programs:

(a) Note manager using file handling:

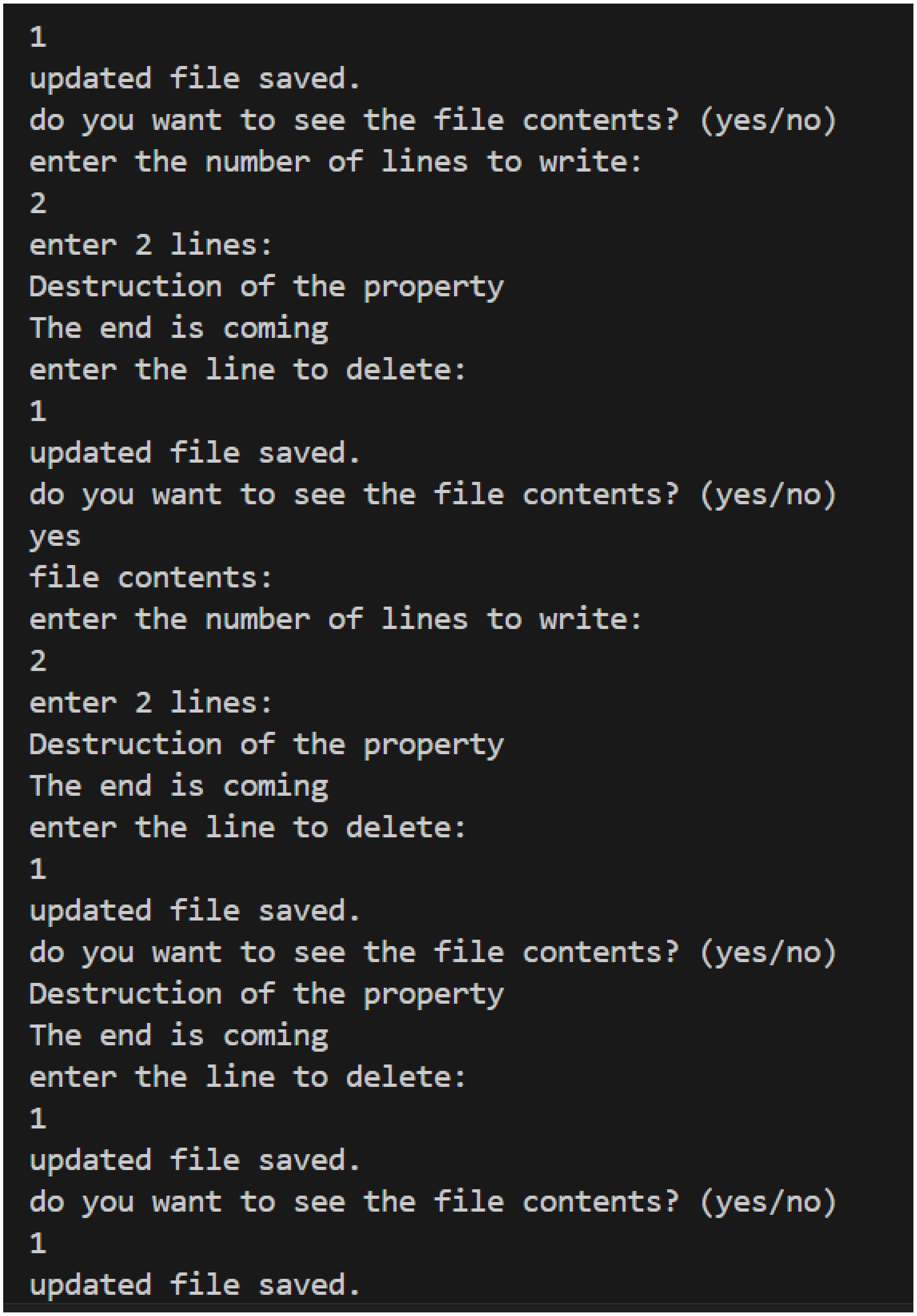
Aim: To let the user write and edit notes in a file. it shows how to use file handling and string simplification to remove a line based on user input. The user is given options – to delete the line which they specify and to view the contents of the file.

Program Code:

```java
import java.io.*;
import java.util.*;

public class notemanager {
    public static void main(String[] args) {
        Scanner scanner = new Scanner(System.in);
        System.out.println("enter the number of lines to 
write:");
        int numLines = scanner.nextInt();
        scanner.nextLine();

        try {
            FileWriter writer = new FileWriter("notes.txt");
            System.out.println("enter " + numLines + "
lines:");
            for (int i = 0; i < numLines; i++) {
                writer.write(scanner.nextLine() + "\n");
            }
            writer.close();
        } catch (IOException e) {
            System.out.println("error: could not write to
file.");
            scanner.close();
            return;
        }

        System.out.println("enter the line to delete:");
        String delline = scanner.nextLine().toLowerCase();
        try {
            File file = new File("notes.txt");
            Scanner fileScanner = new Scanner(file);
            List<String> lines = new ArrayList<>();

            while (fileScanner.hasNextLine()) {
                String line = fileScanner.nextLine();
                if
(!simplify(line).contains(simplify(delline))) {
                    lines.add(line);
                }
            }
            fileScanner.close();

            FileWriter writer = new FileWriter("notes.txt");
            for (String line : lines) {
                writer.write(line + "\n");
            }
            writer.close();
            System.out.println("updated file saved.");
        } catch (IOException e) {
            System.out.println("error: file not found.");
            scanner.close();
            return;
```

```java
        }
        System.out.println("do you want to see the file
contents? (yes/no)");
        String choice = scanner.nextLine().toLowerCase();
        if (choice.equals("yes")) {
            try {
                BufferedReader reader = new
BufferedReader(new FileReader("notes.txt"));
                String line;
                System.out.println("file contents:");
                while ((line = reader.readLine()) != null) {
                    System.out.println(line);
                }
                reader.close();
            } catch (IOException e) {
                System.out.println("error reading the
file.");
            }
        } else {
            System.out.println("okay, not showing the
file.");
        }

        scanner.close();
    }

    private static String simplify(String text) {
        return text.replaceAll("(ing|ed|es|s)$",
"").toLowerCase();
    }
}
```

Output:



```
1
updated file saved.
do you want to see the file contents? (yes/no)
enter the number of lines to write:
2
enter 2 lines:
Destruction of the property
The end is coming
enter the line to delete:
1
updated file saved.
do you want to see the file contents? (yes/no)
yes
file contents:
enter the number of lines to write:
2
enter 2 lines:
Destruction of the property
The end is coming
enter the line to delete:
1
updated file saved.
do you want to see the file contents? (yes/no)
Destruction of the property
The end is coming
enter the line to delete:
1
updated file saved.
do you want to see the file contents? (yes/no)
1
updated file saved.
```

(b) Task storing and management with timestamp:

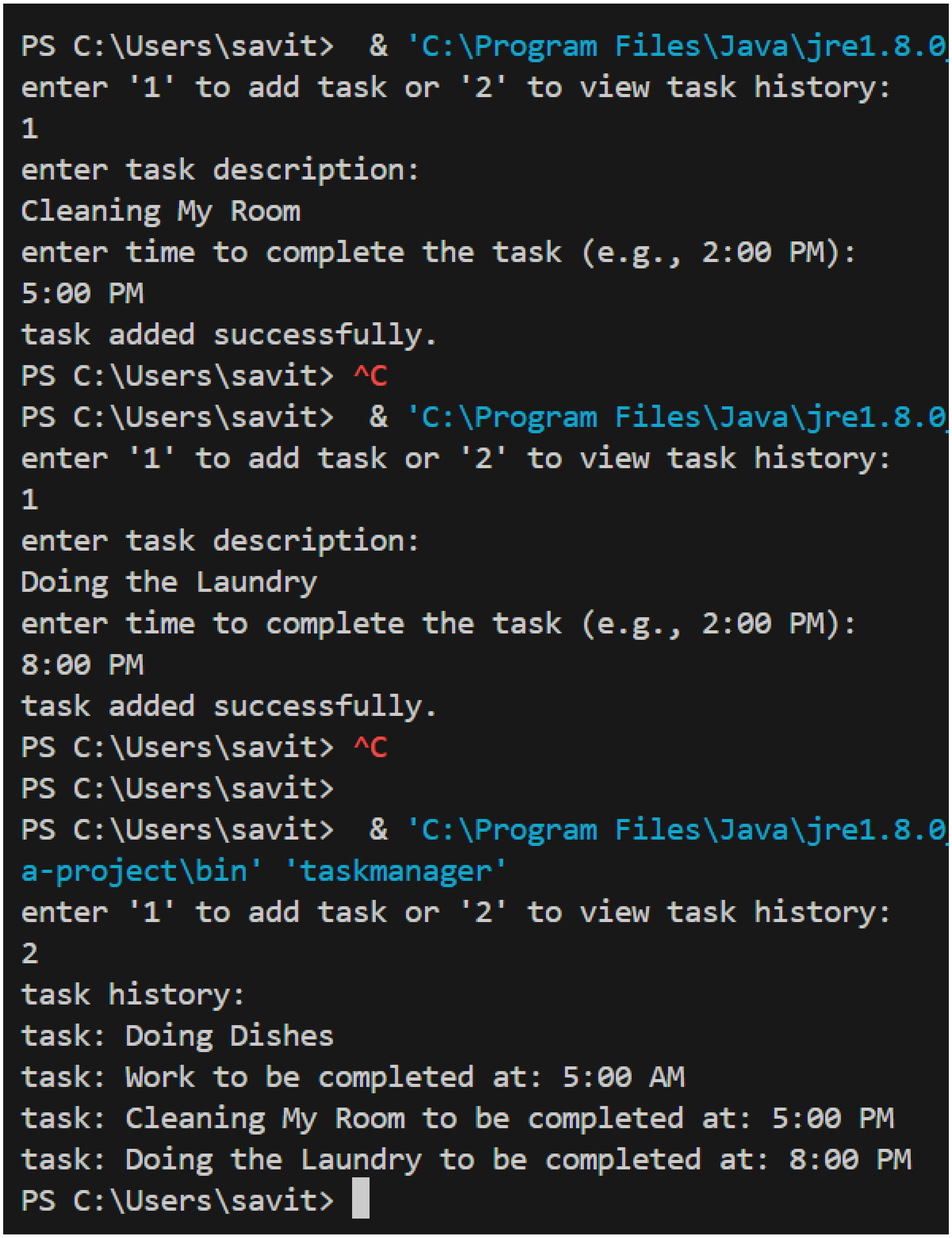
Aim: to allow the user add a task with a time and view saved tasks. it uses file writing and reading to keep a record of tasks.

Program Code:

```java
import java.io.FileWriter;
import java.io.IOException;
import java.io.BufferedReader;
import java.io.FileReader;
import java.util.Scanner;

public class taskmanager {
    public static void main(String[] args) {
        Scanner scanner = new Scanner(System.in);
        System.out.println("enter '1' to add task or '2' to
view task history:");
        int c = scanner.nextInt();
        scanner.nextLine();

        if (c == 1) {
            System.out.println("enter task description:");
            String task = scanner.nextLine();
            System.out.println("enter time to complete the
task (e.g., 2:00 PM):");
            String time = scanner.nextLine();

            try {
                FileWriter writer = new
FileWriter("taskhistory.txt", true);
```

```
                writer.write("task: " + task + " to be
completed at: " + time + "\n");
                writer.close();
                System.out.println("task added
successfully.");
            } catch (IOException e) {
                System.out.println("error: unable to save
task details.");
            }
        } else if (c == 2) {
            try {
                BufferedReader reader = new
BufferedReader(new FileReader("taskhistory.txt"));
                String line;
                System.out.println("task history:");
                while ((line = reader.readLine()) != null) {
                    System.out.println(line);
                }
                reader.close();
            } catch (IOException e) {
                System.out.println("error: unable to read
task history.");
            }
        } else {
            System.out.println("invalid choice.");
        }

        scanner.close();
    }
}
```

Output:



```
PS C:\Users\savit>  & 'C:\Program Files\Java\jre1.8.0
enter '1' to add task or '2' to view task history:
1
enter task description:
Cleaning My Room
enter time to complete the task (e.g., 2:00 PM):
5:00 PM
task added successfully.
PS C:\Users\savit> ^C
PS C:\Users\savit>  & 'C:\Program Files\Java\jre1.8.0
enter '1' to add task or '2' to view task history:
1
enter task description:
Doing the Laundry
enter time to complete the task (e.g., 2:00 PM):
8:00 PM
task added successfully.
PS C:\Users\savit> ^C
PS C:\Users\savit>
PS C:\Users\savit>  & 'C:\Program Files\Java\jre1.8.0
a-project\bin' 'taskmanager'
enter '1' to add task or '2' to view task history:
2
task history:
task: Doing Dishes
task: Work to be completed at: 5:00 AM
task: Cleaning My Room to be completed at: 5:00 PM
task: Doing the Laundry to be completed at: 8:00 PM
PS C:\Users\savit>
```

(c) User login history tracker:

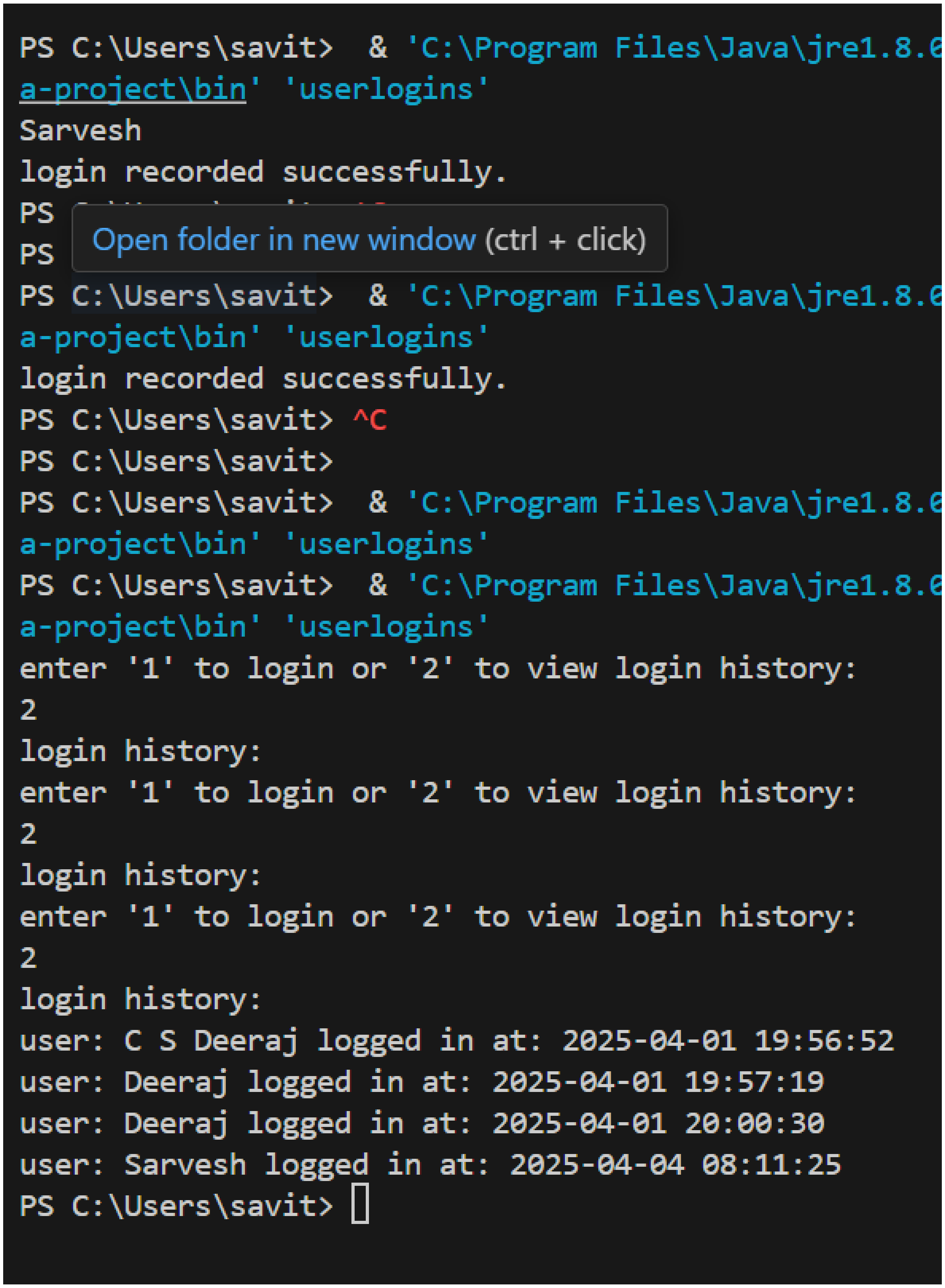
Aim: To let users log in and see past login records with time, using file storage to keep the data.

Program Code:

```
import java.io.FileWriter;
import java.io.IOException;
import java.io.BufferedReader;
import java.io.FileReader;
import java.util.Scanner;
import java.time.LocalDateTime;
import java.time.format.DateTimeFormatter;

public class userlogins {
    public static void main(String[] args) {
        Scanner scanner = new Scanner(System.in);
        System.out.println("enter '1' to login or '2' to 
view login history:");
        int choice = scanner.nextInt();
        scanner.nextLine();

        if (choice == 1) {
            System.out.println("enter username:");
            String username = scanner.nextLine();

            LocalDateTime now = LocalDateTime.now();
            DateTimeFormatter formatter = 
DateTimeFormatter.ofPattern("yyyy-MM-dd HH:mm:ss");
            String timestamp = now.format(formatter);

            try {
                FileWriter writer = new 
FileWriter("loginhistory.txt", true);
```

```java
                writer.write("user: " + username + " logged
in at: " + timestamp + "\n");
                writer.close();
                System.out.println("login recorded
successfully.");
            } catch (IOException e) {
                System.out.println("error: unable to save
login details.");
            }
        } else if (choice == 2) {
            try {
                BufferedReader reader = new
BufferedReader(new FileReader("loginhistory.txt"));
                String line;
                System.out.println("login history:");
                while ((line = reader.readLine()) != null) {
                    System.out.println(line);
                }
                reader.close();
            } catch (IOException e) {
                System.out.println("error: unable to read
login history.");
            }
        } else {
            System.out.println("invalid choice.");
        }

        scanner.close();
    }
}
```

Output:



```
PS C:\Users\savit>  & 'C:\Program Files\Java\jre1.8.0
a-project\bin' 'userlogins'
Sarvesh
login recorded successfully.
PS
PS
PS C:\Users\savit>  & 'C:\Program Files\Java\jre1.8.0
a-project\bin' 'userlogins'
login recorded successfully.
PS C:\Users\savit> ^C
PS C:\Users\savit>
PS C:\Users\savit>  & 'C:\Program Files\Java\jre1.8.0
a-project\bin' 'userlogins'
PS C:\Users\savit>  & 'C:\Program Files\Java\jre1.8.0
a-project\bin' 'userlogins'
enter '1' to login or '2' to view login history:
2
login history:
enter '1' to login or '2' to view login history:
2
login history:
enter '1' to login or '2' to view login history:
2
login history:
user: C S Deeraj logged in at: 2025-04-01 19:56:52
user: Deeraj logged in at: 2025-04-01 19:57:19
user: Deeraj logged in at: 2025-04-01 20:00:30
user: Sarvesh logged in at: 2025-04-04 08:11:25
PS C:\Users\savit> 
```

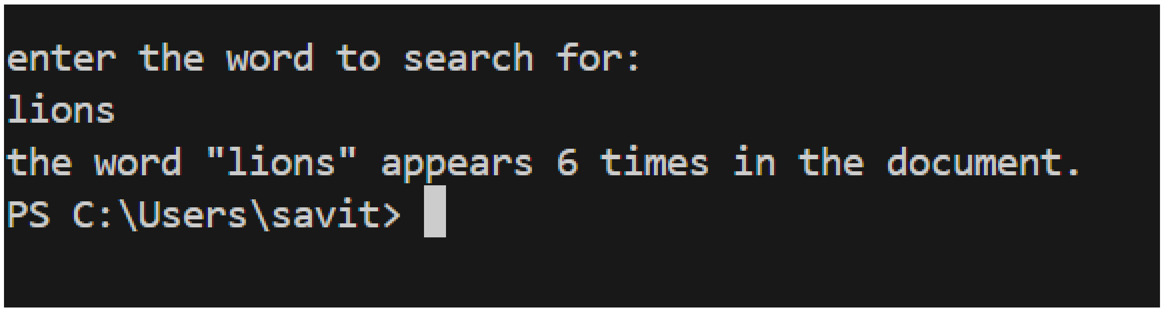
(d) Word search in a document:

Aim: to check how many times a word appears in a text file by writing to the file first and then scanning through it.

Program Code:

```
import java.io.File;
import java.io.FileWriter;
import java.io.IOException;
import java.util.Scanner;

public class wordfrequency {
    public static void main(String[] args) {
        try {
            FileWriter writer = new
FileWriter("document.txt");
            writer.write("the lion is known as the king of
the jungle. the lion is a powerful predator, and the lion
hunts in groups called prides. the lion's roar can be heard
from miles away. lions live in the wild, and lions are also
found in some reserves. lions are strong, and lions have
sharp claws and strong teeth. lions are truly majestic
animals.");
            writer.close();
        } catch (IOException e) {
            System.out.println("error: could not create
file.");
            return;
        }

        Scanner scanner = new Scanner(System.in);
        System.out.println("enter the word to search for:");
        String sw = scanner.nextLine().toLowerCase();
        scanner.close();

        int count = 0;

        try {
            File file = new File("document.txt");
            Scanner fileScanner = new Scanner(file);
            while (fileScanner.hasNext()) {
                String word = 
fileScanner.next().replaceAll("[^a-zA-Z]", 
"").toLowerCase();
                if (word.equals(sw)) {
                    count++;
                }
            }
            fileScanner.close();
            System.out.println("the word \"" + sw + "\" 
appears " + count + " times in the document.");
        } catch (IOException e) {
            System.out.println("error: file not found.");
        }
    }
}
```

Output:



```
enter the word to search for:
lions
the word "lions" appears 6 times in the document.
PS C:\Users\savit> 
```