DAY 6

1) **Compass Directions**

```
import java.util.Scanner;
enum CardinalDirections {
   NORTH, SOUTH, EAST, WEST
}
public class Compass {
   public static void main(String[] args) {
      Scanner sc = new Scanner(System.in);
      String input = sc.next().toUpperCase();
      try {
         CardinalDirections dir = CardinalDirections.valueOf(input);
         switch (dir) {
            case NORTH -> System.out.println("Heading North");
            case SOUTH -> System.out.println("Move south");
            case EAST -> System.out.println("Move east");
            case WEST -> System.out.println("Move west");
         }
      } catch (IllegalArgumentException e) {
         System.out.println("Invalid direction");
      }
   }
}
```

Input:

```
          NORTH
```

Output:

```
        Heading North
```

2) **Object Casting**

```
class Creature {
   void makeSound() {
      System.out.println(" Sound by the Creature");
   }
}
class Tiger extends Creature {
   void howls() {
      System.out.println("Woof!");
   }
   void hunts() {
      System.out.println("Tiger hunts for its food");
   }
}
public class CastingDemo {
   public static void main(String[] args) {
      Tiger d = new Tiger();
      Creature a = d;
      ((Tiger) a).hunts();
      a.makeSound();
   }
}
```

Output:

```
Tiger hunts for its food
Sound by the Creature
```

3) **Compound**

```
public class MathAssignment {
   public static void main(String[] args) {
      int x = 5;
      x *= 2.5;
      System.out.println(x);
   }
}
```
Output:
12

4) **Days of the Week**

```
import java.util.Scanner;
enum WorkDaysAndWeekends {
    MONDAY, TUESDAY, WEDNESDAY, THURSDAY, FRIDAY, SATURDAY, SUNDAY
}
public class CalendarDay {
    public static void main(String[] args) {
        Scanner sc = new Scanner(System.in);
        String dayName = sc.next().toUpperCase();
        try {
            WorkDaysAndWeekends day = WorkDaysAndWeekends.valueOf(dayName);
            if (day == WorkDaysAndWeekends.FRIDAY) {
                System.out.println("Weekend");
            } else {
                System.out.println("Weekday");
            }
            System.out.println(day.ordinal());
        } catch (IllegalArgumentException e) {
            System.out.println("Invalid day");
        }
    }
}
```
Input:
FRIDAY

5) **Difficulty Level & Game Setup**

```
enum GameDifficulty { EASY, MEDIUM, HARD }
class GameSetup {
    GameSetup(GameDifficulty diff) {
        switch (diff) {
            case EASY -> System.out.println("8000 bullets");
            case MEDIUM -> System.out.println("1000 bullets");
            case HARD -> System.out.println("100 bullets");
        }
    }
}
public class GameDemo {
    public static void main(String[] args) {
        new GameSetup(GameDifficulty.EASY);
        new GameSetup(GameDifficulty.HARD);
    }
}
```
Output:
8000 bullets
100 bullets

6) **Custom Exception**

```
class EvenNumberException extends Exception {
    public EvenNumberException(String message) { super(message); }
}
class NumberValidator {
    public static void checkEven(int n) throws EvenNumberException {
        if (n % 2 == 0) {
            throw new EvenNumberException("This is an even number: " + n);
        } else {
            System.out.println("This is an odd number: " + n);
        }
    }
}
public class NumberChecker {
    public static void main(String[] args) {
        try {
            NumberValidator.checkEven(7);
        } catch (EvenNumberException e) {
            System.out.println(e.getMessage());
        }
        try {
            NumberValidator.checkEven(8);
        } catch (EvenNumberException e) {
            System.out.println(e.getMessage());
        }
    }
}
```

Output:

This is an odd number: 7
This is an even number: 8

7) **Multiple Catches**

```
import java.io.*;
public class FileUtility {
    public static void main(String[] args) {
        String filename = "Gamefile.txt";
        try {
            readAndPrintFile(filename);
        } catch (FileNotFoundException e) {
            System.out.println("The file '" + filename + "' was not found.");
        } catch (IOException e) {
            System.out.println("There was an issue reading the file: " + e.getMessage());
        } finally {
            System.out.println("File operation completed.");
        }
    }

    public static void readAndPrintFile(String filename) throws FileNotFoundException, IOException {
        BufferedReader br = new BufferedReader(new FileReader(filename));
        String line = br.readLine();
        System.out.println(line);
        br.close();
    }
}
```

Output:

The file 'Gamefile.txt' was not found.
File operation completed.

8) **Traffic Light**

```
interface TrafficState { TrafficState next(); }
enum LightColor implements TrafficState {
    RED {
        public TrafficState next() { return GREEN; }
    },
    GREEN {
        public TrafficState next() { return YELLOW; }
    },
    YELLOW {
        public TrafficState next() { return RED; }
    }
}
public class LightCycle {
    public static void main(String[] args) {
        TrafficState state = LightColor.RED;
        for (int i = 0; i < 4; i++) {
            System.out.println(state);
            state = state.next();
        }
    }
}
```

Output:

RED

```
GREEN
YELLOW
RED
```

9) Shape Area Calculator

```
enum GeometricShape {
    CIRCLE {
        double area(double... params) { return Math.PI * params[0] * params[0]; }
    },
    SQUARE {
        double area(double... params) { return params[0] * params[0]; }
    },
    RECTANGLE {
        double area(double... params) { return params[0] * params[1]; }
    },
    TRIANGLE {
        double area(double... params) { return 0.5 * params[0] * params[1]; }
    },
    HEXAGON {
        double area(double... params) { return (3 * Math.sqrt(3) / 2) * params[0] * params[0]; }
    };
    abstract double area(double... params);
}
public class GeometryCalculator {
    public static void main(String[] args) {
        System.out.println("Circle: " + GeometricShape.CIRCLE.area(2));
        System.out.println("Square: " + GeometricShape.SQUARE.area(7));
        System.out.println("Rectangle: " + GeometricShape.RECTANGLE.area(9, 6));
        System.out.println("Triangle: " + GeometricShape.TRIANGLE.area(3, 8));
        System.out.println("Hexagon: " + GeometricShape.HEXAGON.area(11));
    }
}
```

Output:

```
Circle: 12.566370614359172
Square: 49.0
Rectangle: 54.0
Triangle: 12.0
Hexagon: 317.5689104034873
```

10) Multiple Exceptions

```
import java.io.*;
public class ExceptionChain {
    public static void main(String[] args) {
        try {
            BufferedReader br = new BufferedReader(new FileReader("input.txt"));
            String line = br.readLine();
            int num = Integer.parseInt(line);
            System.out.println(num);
            br.close();
        } catch (FileNotFoundException e) {
            System.out.println("The file was not found.");
        } catch (IOException e) {
            System.out.println("An I/O error occurred.");
        } catch (NumberFormatException e) {
            System.out.println("The file contains an invalid number format.");
        } finally {
            System.out.println("The program has finished its execution.");
        }
    }
}
```

Output :

```
The file contains an invalid format of number.
The program has completed its execution.
```

11) **Knowledge Level**

```
enum Skill {
    BEGINNER, ADVANCED, PROFESSIONAL, MASTER;
    static SkillfromScore(int score) {
        if (score <= 5) return BEGINNER;
        else if (score <= 10) return ADVANCED;
        else if (score <= 15) return PROFESSIONAL;
        else return MASTER;
    }
}
public class SkillApp {
    public static void main(String[] args) {
        System.out.println(Skill.fromScore(2));
        System.out.println(Skill.fromScore(9));
        System.out.println(Skill.fromScore(14));
        System.out.println(Skill.fromScore(50));
    }
}
```

Output:

```
BEGINNER
ADVANCED
PROFESSIONAL
MASTER
```

12) **Division**

```
public class HandelError {
    public static void main(String[] args) {
        try {
            int a = 18 / 0;
        } catch (ArithmeticException e) {
            System.out.println("Cannot perform division by zero.");
        } finally {
            System.out.println("Operation finished.");
        }
    }
}
```

Output:

```
Cannot perform division by zero.
Operation finished.
```

13) **Priority Levels**

```
enum TaskPriorityLevel {
    LOW(1), MEDIUM(2), HIGH(3), CRITICAL(4);
    int code;
    TaskPriorityLevel(int code) { this.code = code; }
    boolean isUrgent() { return code >= 3; }
}
public class TaskPriority {
    public static void main(String[] args) {
        for (TaskPriorityLevel p : TaskPriorityLevel.values()) {
            System.out.println(p + " Code: " + p.code + " Urgent: " + p.isUrgent());
        }
    }
}
```

Output:

```
LOW Code: 1 Urgent: false
MEDIUM Code: 2 Urgent: false
HIGH Code: 3 Urgent: true
CRITICAL Code: 4 Urgent: true
```

14) **Temperature Converter**

```
import java.util.Scanner;
public class WeatherConverter {
    public static void main(String[] args) {
        Scanner sc = new Scanner(System.in);
        double celsius = sc.nextDouble();
        double fahrenheit = (celsius * 1.8) + 32;
        int truncated = (int) fahrenheit;
        System.out.println(fahrenheit);
        System.out.println(truncated);
    }
}
```

Input:

45

output:

113.0

113