INTERFACE

Exercise:

```
// Interface defining constants
interface Constants {
    int TOTAL_MAX_MARKS = 8000;
    int INSTITUTE_A_COURSE_MARKS = 900;
    int INSTITUTE_A_GRACE_MARKS = 100;
    int INSTITUTE_B_COURSE_MARKS = 1000;
}

// Interface for calculating percentage
interface Percentage {
    double calcPercentage();
}

// Class for Intern in Institute A
class Intern implements Percentage, Constants {
    protected int marksSecured;
    protected int graceMarks;

    public Intern(int marksSecured, int graceMarks) {
        this.marksSecured = marksSecured;
        this.graceMarks = graceMarks;
    }

    @Override
    public double calcPercentage() {
        if (marksSecured <= TOTAL_MAX_MARKS - INSTITUTE_A_GRACE_MARKS) {
            int totalMarks = marksSecured + graceMarks;
            return (double) totalMarks / TOTAL_MAX_MARKS * 100;
        } else {
            System.out.println("Please enter the correct marks");
            return -1; // or throw an exception as per your requirement
        }
    }
}

// Class for Trainee in Institute B
class Trainee implements Percentage, Constants {
    protected int marksSecured;
```

```
    public Trainee(int marksSecured) {
        this.marksSecured = marksSecured;
    }

    @Override
    public double calcPercentage() {
        int totalMarks = marksSecured;
        return (double) totalMarks / TOTAL_MAX_MARKS * 100;
    }
}

// PercentageCalculator class to demonstrate the usage
public class PercentageCalculator {
    public static void main(String[] args) {
        // Example usage for Intern
        Intern intern1 = new Intern(5000, 500);
        System.out.println("Input (For Intern):");
        System.out.println("Marks Secured: " + intern1.marksSecured);
        System.out.println("Grace Marks: " + intern1.graceMarks);
        System.out.println("Output:");
        System.out.println("The total aggregate percentage secured is " +
intern1.calcPercentage());

        // Example usage for Trainee
        Trainee trainee1 = new Trainee(6000);
        System.out.println("\nInput (For Trainee):");
        System.out.println("Marks Secured: " + trainee1.marksSecured);
        System.out.println("Output:");
        System.out.println("The total aggregate percentage secured is " +
trainee1.calcPercentage());

        // Additional example with invalid marks for Intern
        Intern intern2 = new Intern(8000, 500);
        System.out.println("\nInput (For Intern):");
        System.out.println("Marks Secured: " + intern2.marksSecured);
        System.out.println("Grace Marks: " + intern2.graceMarks);
        System.out.println("Output:");
        System.out.println("The total aggregate percentage secured is " +
intern2.calcPercentage());
```

```
    }
}
```

Output:

```
The total aggregate percentage secured is 68.75

Input (For Trainee):
Marks Secured: 6000
Output:
The total aggregate percentage secured is 75.0

Input (For Intern):
Marks Secured: 8000
Grace Marks: 500
Output:
Please enter the correct marks
The total aggregate percentage secured is -1.0
```

PACKAGE

Exercise:

```
// Package customer
package customer;

// Customer class
public class Customer {
    protected int customerId;
    private String address;
    protected String phoneNo;
    protected String customerName;

    public Customer(int customerId, String address, String phoneNo, String
customerName) {
        this.customerId = customerId;
        this.address = address;
        this.phoneNo = phoneNo;
        this.customerName = customerName;
    }

    public void displayCustomerDetails() {
        System.out.println("Customer Id: " + customerId);
        System.out.println("Customer Name: " + customerName);
```

```
        System.out.println("Contact Nos: " + phoneNo);
        System.out.println("Address:\n" + address);
    }
}

// Sub-package customer.typeofcustomer
package customer.typeofcustomer;

import customer.Customer; // Importing Customer class from the customer
package

// RegularCustomer class
public class RegularCustomer extends Customer {
    public RegularCustomer(int customerId, String address, String phoneNo,
String customerName) {
        super(customerId, address, phoneNo, customerName);
    }
}

// Package purchase
package purchase;

import customer.Customer;

// PurchaseBill class
public class PurchaseBill {
    private int billId;
    private double processingCharge;
    private double discount;

    public PurchaseBill(int billId, double processingCharge, double
discount) {
        this.billId = billId;
        this.processingCharge = processingCharge;
        this.discount = discount;
    }

    public void calculateBillAmount(Customer customer, double billAmount)
{
        double finalAmount = billAmount + processingCharge - discount;
```

```
        System.out.println("Easy Shop Retail Store Bill");
        System.out.println("Bill Id: " + billId);
        customer.displayCustomerDetails();
        System.out.println("Discount: " + discount);
        System.out.println("Final bill amount to be paid: Rs." +
finalAmount);
        System.out.println("Thank You!!! Visit Again");
    }
}

// Application class (formerly Main)
import customer.typeofcustomer.RegularCustomer;
import purchase.PurchaseBill;
import customer.Customer;

public class Application {
    public static void main(String[] args) {
        // Creating a RegularCustomer object
        RegularCustomer customer = new RegularCustomer(
                1001, "No. 333, ABC street, Mysore, Karnataka, 570001",
                "9962788712 9886124566 9496781256", "John");

        // Creating a PurchaseBill object and calculating the bill amount
        PurchaseBill bill = new PurchaseBill(5001, 150, 10.0);
        bill.calculateBillAmount(customer, 2000);
    }
}
```

Output:

```
Easy Shop Retail Store Bill
Bill Id: 5001
Customer Id: 1001
Customer Name: John
Contact Nos: 9962788712 9886124566 9496781256
Address:
No. 333, ABC street, Mysore, Karnataka, 570001
Discount: 10.0
Final bill amount to be paid: Rs.2140.0
Thank You!!! Visit Again
```

ACCESS MODIFIERS

Exercise:

```java
// InfyTV class
public class InfyTV {
    // Private member variables
    private String photographer;
    private String newsReporter;
    private String correspondent;

    // Getter methods to access private member variables
    public String getPhotographer() {
        return photographer;
    }

    public String getNewsReporter() {
        return newsReporter;
    }

    public String getCorrespondent() {
        return correspondent;
    }

    // Setter methods to initialize private member variables
    public void setPhotographer(String photographer) {
        this.photographer = photographer;
    }

    public void setNewsReporter(String newsReporter) {
        this.newsReporter = newsReporter;
    }

    public void setCorrespondent(String correspondent) {
        this.correspondent = correspondent;
    }

    // Public method to display the documentary details
    public void documentaryFilm() {
        System.out.println("A hundred years ago there were 100,000 tigers
in the world. " +
                "Today there are as few as 3,200. Why are tigers
disappearing?......");
        System.out.println("by Correspondent: " + correspondent);
```

```
        System.out.println("Photographer: " + photographer);
        System.out.println("newsReporter: " + newsReporter);
    }
}

// Tester class with the main method
public class Tester {
    public static void main(String[] args) {
        // Create an object of InfyTV class
        InfyTV infyTV = new InfyTV();

        // Try accessing private member variables directly (cannot access)
        //infyTV.photographer = "Joshua";  // This will give a compilation
error
        //infyTV.newsReporter = "Hudson";  // This will give a compilation
error
        //infyTV.correspondent = "Kimberely";  // This will give a
compilation error

        // Access private members using respective setters
        infyTV.setPhotographer("Joshua");
        infyTV.setNewsReporter("Hudson");
        infyTV.setCorrespondent("Kimberely");

        // Display the documentary details
        infyTV.documentaryFilm();
    }
}
```

Output:

```
A hundred years ago there were 100,000 tigers in the world. Today there are as few as 3,200. Why are tigers disappearing?.....
by Correspondent: Kimberely
Photographer: Joshua
newsReporter: Hudson
```

JAVA LIBRARIES

Exercise:

```
public class RegistrationTester {
    public static void main(String[] args) {
```

```
        // Create a student registration object to hold values for student
with previously allotted roll number
        StudentRegistration student1 = new StudentRegistration("Peter",
23, 5001);
        student1.generateRollNumber();

        // Create a student registration object to hold values for student
pending roll number allotment
        StudentRegistration student2 = new StudentRegistration("Peter",
24, 5003);
        student2.generateRollNumber();

        // Compare both objects and display results
        if (student1.equals(student2)) {
            System.out.println("Roll number already generated for the
student!!");
        } else {
            System.out.println("Student Details");
            System.out.println("Student Name: " +
student2.getStudentName());
            System.out.println("Admission Number: " +
student2.getAdmissionNumber());
            System.out.println("Roll Number: " +
student2.getRollNumber());
        }
    }
}

public class StudentRegistration {
    private String studentName;
    private double age;
    private int admissionNumber;
    private int rollNumber;
    private static int counter = 1001;

    public StudentRegistration(String studentName, double age, int
admissionNumber) {
        this.studentName = studentName;
        this.age = age;
        this.admissionNumber = admissionNumber;
```

```
    }

    public String getStudentName() {
        return studentName;
    }

    public double getAge() {
        return age;
    }

    public int getAdmissionNumber() {
        return admissionNumber;
    }

    public int getRollNumber() {
        return rollNumber;
    }

    public void generateRollNumber() {
        if (rollNumber == 0) {
            rollNumber = counter++;
        }
    }

    @Override
    public int hashCode() {
        final int prime = 31;
        int result = 1;
        result = prime * result + admissionNumber;
        result = prime * result + ((studentName == null) ? 0 :
studentName.hashCode());
        return result;
    }

    @Override
    public boolean equals(Object obj) {
        if (this == obj)
            return true;
        if (obj == null)
            return false;
```

```
        if (getClass() != obj.getClass())
            return false;
        StudentRegistration other = (StudentRegistration) obj;
        if (admissionNumber != other.admissionNumber)
            return false;
        if (studentName == null) {
            if (other.studentName != null)
                return false;
        } else if (!studentName.equals(other.studentName))
            return false;
        return true;
    }
}
```

Output:

```
Student Details
Student Name: Peter
Admission Number: 5003
Roll Number: 1002
```

Exercise:

```
public class LocationChanger {
    private String location;
    private String name;

    public String getLocation() {
        return location;
    }

    public void setLocation(String location) {
        this.location = location;
    }

    public String getName() {
        return name;
    }

    public void setName(String name) {
        this.name = name;
```

```
    }

    public void checkCity() {
        if (location.contains("Delhi")) {
            System.out.println("Welcome to Infy Mysore Delhites!");
        } else if (location.contains("Trivandrum")) {
            System.out.println("Welcome to MyDC people of Trinfy!!");
        } else if (location.contains("Bhubaneshwar")) {
            System.out.println("You came a long way down South! We welcome
you!");
        } else {
            System.out.println("Oops your city name is not listed! Your
location remains the same!");
        }
    }

    public void editAddress() {
        if (location.contains("STP")) {
            System.out.println("Your location has been changed from STP to
SEZ");
        } else if (location.contains("SEZ")) {
            System.out.println("Your location remains the same!");
        }
    }

    public void welcomeEmployee() {
        String[] parts = name.split(" ");
        System.out.println("Hello " + parts[0]);
    }
}




public class Tester {
    public static void main(String[] args) {
        // Create two instances of LocationChanger and initialize with
given values
        LocationChanger employee1 = new LocationChanger();
```

```
        employee1.setName("Annabelle Michael");
        employee1.setLocation("BL003, Delhi, STP");

        LocationChanger employee2 = new LocationChanger();
        employee2.setName("Jissele James");
        employee2.setLocation("FL091, Pune, SEZ");

        // Invoke methods: welcomeEmployee(), checkCity(), and
editAddress()
        employee1.welcomeEmployee();
        employee1.checkCity();
        employee1.editAddress();

        employee2.welcomeEmployee();
        employee2.checkCity();
        employee2.editAddress();
    }
}
```

Output:

```
Hello Annabelle
Welcome to Infy Mysore Delhites!
Your location has been changed from STP to SEZ
Hello Jissele
Oops your city name is not listed! Your location remains the same!
Your location remains the same!
```

Exercise:

```
public class PalindromeChecker {
    public static boolean isPalindrome(int number) {
        String numStr = String.valueOf(number); // Convert the number to a
String
        StringBuilder reversed = new StringBuilder(numStr).reverse(); //
Reverse the String

        // Compare original and reversed strings ignoring case
        return numStr.equals(reversed.toString());
    }

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

```
        int number = 45687;
        if (isPalindrome(number)) {
            System.out.println(number + " is a palindrome");
        } else {
            System.out.println(number + " is not a palindrome");
        }
    }
}
```

Output:

```
45687 is not a palindrome
```

EXCEPTIONS

Exercise:

```
public class TryMathCube {
    public int cube(int n) {
        return n * n * n;
    }
    public static void main(String[] args) {
        TryMathCube tc = new TryMathCube();
        try {
            if (args.length == 0) {
                throw new IllegalArgumentException("Please provide an integer
as a command-line argument.");
            }
            int num = Integer.parseInt(args[0]);
            System.out.println(tc.cube(num));
        } catch (NumberFormatException e) {
            System.out.println("Invalid input. Please provide a valid
integer.");
        } catch (IllegalArgumentException e) {
            System.out.println(e.getMessage());
        }
    }
}
```

Output:

```
Please provide an integer as a command-line argument.
```

Exercise:

```
public class Employee {
    private String empName;
    private int empAge;
    private double empSalary;

    public Employee(String empName, int empAge, double empSalary) {
        this.empName = empName;
        this.empAge = empAge;
        this.empSalary = empSalary;
    }

    // Getters and setters for employee details (empName, empAge,
empSalary)

    public String getEmpName() {
        return empName;
    }

    public void setEmpName(String empName) {
        this.empName = empName;
    }

    public int getEmpAge() {
        return empAge;
    }

    public void setEmpAge(int empAge) {
        this.empAge = empAge;
    }

    public double getEmpSalary() {
        return empSalary;
    }

    public void setEmpSalary(double empSalary) {
        this.empSalary = empSalary;
    }
}

public class EmpSalaryException extends Exception {
```

```
    public EmpSalaryException(String message) {
        super(message);
    }
}


public class EmployeeService {
    public void checkEmployeeSalary(Employee emp) {
        try {
            if (emp.getEmpSalary() < 1000) {
                throw new EmpSalaryException("Salary less than 1000");
            }
        } catch (EmpSalaryException e) {
            System.out.println("Employee Name: " + emp.getEmpName() + "
has salary less than 1000");
        }
    }

    public static void main(String[] args) {
        Employee e1 = new Employee("Joe Smith", 20, 5345);
        Employee e2 = new Employee("Lewis Jane", 21, 1345);
        Employee e3 = new Employee("Larry Page", 22, 245);
        Employee e4 = new Employee("Smith James", 23, 945);
        Employee e5 = new Employee("Elvis George", 25, 1425);
        EmployeeService empSer = new EmployeeService();
        Employee[] empArray = { e1, e2, e3, e4, e5 };

        for (Employee employee : empArray) {
            empSer.checkEmployeeSalary(employee);
        }
    }
}
```

Output:

```
Employee Name: Larry Page has salary less than 1000
Employee Name: Smith James has salary less than 1000
```

GENERICS

Exercise:

```java
public class SimpleGeneric<T> {

    private T obj;

    public SimpleGeneric(T obj) {
        this.obj = obj;
    }

    public void showObjectType() {
        System.out.println("Object Type is " + obj.getClass().getName());
    }

    public static void main(String[] args) {
        SimpleGeneric<String> genClass = new SimpleGeneric<>("Input");
        genClass.showObjectType();
    }
}
```

Output:

```
Object Type is java.lang.String
```