lab 1

```
// Define the Account class
public class Account
// Private instance variables
private String accountNumber;
private double balance;

// Parameterized constructor with validation
public Account(String accountNumber, double balance)
// Validate accountNumber
if (accountNumber == null || accountNumber.isEmpty())
// Print error message if accountNumber is null or empty
System.err.println("Error: Account number cannot be null or empty.");
return;
```

```
// Validate balance
if (balance < 0)
// Print error message if balance is negative
System.err.println("Error: Balance cannot
be negative.");
return;

// Initialize accountNumber with the
provided parameter
this.accountNumber = accountNumber;
// Initialize balance with the provided
parameter
this.balance = balance;


// Main method to test the Account class
public static void main(String[] args)
// Test with valid data
```

```
Account account1 = new
Account("12340009", 1000.00);
System.out.println("Account 1 Number: " +
account1.accountNumber);
System.out.println("Account 1 Balance: " +
account1.balance);

// Test with invalid accountNumber
Account account2 = new Account("",
400.00);

// Test with invalid balance
Account account3 = new
Account("1230000873", -200.00);



// Define a public class named Animal
public class Animal
// Define a public method named
```

makeSound

```
public void makeSound()
// Print "The animal makes a sound." to the console
System.out.println("The animal makes a sound.");



public class BankAccount
// Private field to store the account number
private String accountNumber;

// Private field to store the balance
private double balance;

// Constructor to initialize account number and balance
public BankAccount(String accountNumber, double balance)
```

```
this.accountNumber = accountNumber;
this.balance = balance;


// Method to deposit an amount into the account
public void deposit(double amount)
// Increase the balance by the deposit amount
balance += amount;


// Method to withdraw an amount from the account
public void withdraw(double amount)
// Check if the balance is sufficient for the withdrawal
if (balance >= amount)
// Decrease the balance by the withdrawal
```

```
amount
balance -= amount;
  else
// Print a message if the balance is
insufficient
System.out.println("Insufficient balance");



// Method to get the current balance
public double getBalance()
// Return the current balance
return balance;


// Define the Cat class
public class Cat
// Private instance variables
```

```
private String name;
private int age;
// Default constructor
public Cat()
// Initialize name to "Unknown"
this.name = "Unknown";
// Initialize age to 0
this.age = 0;

// Getter for name
public String getName()
return name;

// Getter for age
public int getAge()
return age;

// Main method to test the Cat class
```

```
public static void main(String[] args)
// Create a new Cat object using the default constructor
Cat myCat = new Cat();
// Use the getter methods to access private variables
System.out.println("Cat's Name: " + myCat.getName());
System.out.println("Cat's Age: " + myCat.getAge());


// Define the Dog class
public class Dog
// Private instance variables
private String name;
private String color;

// Parameterized constructor
```

```
public Dog(String name, String color)
// Initialize name with the provided
parameter
this.name = name;
// Initialize color with the provided
parameter
this.color = color;



// Main method to test the Dog class
public static void main(String[] args)
// Create a new Dog object using the
parameterized constructor
Dog myDog = new Dog("Bailey", "Black");
// Print the values of the instance variables
System.out.println("Dog's Name: " +
myDog.name);
System.out.println("Dog's Color: " +
myDog.color);
```

```
// Employee.java
// Parent class Employee
public class Employee

// Private field to store the salary of the employee
private int salary;

// Constructor to initialize the salary of the employee
public Employee(int salary)
this.salary = salary;


// Method to simulate the employee working
public void work()
// Print a message indicating the employee
```

```
is working
System.out.println("working as an
employee!");


// Getter method to retrieve the salary of the
employee
public int getSalary()
return salary;


// Define the Rectangle class
public class Rectangle
// Private instance variables
private double length;
private double width;

// Parameterized constructor
public Rectangle(double length, double
width)
// Initialize length with the provided
parameter
this.length = length;
// Initialize width with the provided
parameter
this.width = width;


// Copy constructor
public Rectangle(Rectangle rectangle)

this.length = rectangle.length;
// Initialize width with the width of the
provided rectangle object
this.width = rectangle.width;


// Main method to test the Rectangle class
```

```
public static void main(String[] args)
// Create a new Rectangle object using the
parameterized constructor
Rectangle rect1 = new Rectangle(12.5,
4.5);
// Print the values of the instance variables
for rect1
System.out.println("Rectangle 1 Length: " +
rect1.length);
System.out.println("Rectangle 1 Width: " +
rect1.width);

// Create a new Rectangle object using the
copy constructor
Rectangle rect2 = new Rectangle(rect1);
// Print the values of the instance variables
for rect2
System.out.println("Rectangle 2 Length: " +
rect2.length);
System.out.println("Rectangle 2 Width: " +
rect2.width);


// Define the parent class Shape
public class Shape
// Define a public method named getArea that returns a double
public double getArea()
// Return 0.0 as the default area
return 0.0;


// Define the Student class
public class Student
// Private instance variables
private int studentId;
private String studentName;
private String grade;
```

```
// Default constructor
public Student()
// Call the parameterized constructor with default values
this(0, "Unknown", "None");



// Parameterized constructor
public Student(int studentId, String studentName, String grade)
// Initialize studentId with the provided parameter
this.studentId = studentId;
// Initialize studentName with the provided parameter
this.studentName = studentName;
// Initialize grade with the provided parameter
```

```
this.grade = grade;



// Main method to test the Student class
public static void main(String[] args)
// Create a new Student object using the
default constructor
Student student1 = new Student();
// Print the values of the instance variables
for student1
System.out.println("Student1 ID: " +
student1.studentId);
System.out.println("Student1 Name: " +
student1.studentName);
System.out.println("Student1 Grade: " +
student1.grade);

// Create a new Student object using the
parameterized constructor
```

```
Student student2 = new Student(101,
"Cullen", "A");
// Print the values of the instance variables
for student2
System.out.println("Student2 ID: " +
student2.studentId);
System.out.println("Student2 Name: " +
student2.studentName);
System.out.println("Student2 Grade: " +
student2.grade);


// Define the parent class Vehicle
class Vehicle
// Define a public method named drive
public void drive()
// Print "Repairing a vehicle" to the console
System.out.println("Repairing a vehicle");
```