Enterprise Java Lab - Assignment 4 Solutions

1. Reverse String using StringBuilder

```
public class ReverseString {
    public static void main(String[] args) {
        String str = "Hello World";
        System.out.println("Reversed: " + new StringBuilder(str).reverse());
    }
}
```

2. Count Vowels and Consonants

```
public class VowelConsonant {
    public static void main(String[] args) {
        String str = "Hello World".toLowerCase();
        int v = 0, c = 0;
        for (char ch : str.toCharArray()) {
            if (ch >= 'a' && ch <= 'z') {
                if ("aeiou".indexOf(ch) != -1) v++;
                else c++;
            }
        }
        System.out.println("Vowels: " + v + ", Consonants: " + c);
    }
}
```

3. Check Anagrams

```
import java.util.Arrays;

public class Anagram {
    public static void main(String[] args) {
        String s1 = "listen", s2 = "silent";
        char[] a1 = s1.toCharArray(), a2 = s2.toCharArray();
        Arrays.sort(a1); Arrays.sort(a2);
        System.out.println(Arrays.equals(a1, a2) ? "Anagrams" : "Not Anagrams");
    }
}
```

4. Abstract Shape

```
abstract class Shape {
    abstract double area();
}
class Circle extends Shape {
    double r = 3;
    double area() { return Math.PI * r * r; }
}
class Rectangle extends Shape {
    double l = 4, w = 5;
    double area() { return l * w; }
}
public class ShapeTest {
    public static void main(String[] args) {
        Shape s1 = new Circle();
        Shape s2 = new Rectangle();
        System.out.println("Circle Area: " + s1.area());
        System.out.println("Rectangle Area: " + s2.area());
    }
}
```

5. Abstract Vehicle

```
abstract class Vehicle {
    abstract void start();
}
class Car extends Vehicle {
    void start() { System.out.println("Car starts with key"); }
}
class Bike extends Vehicle {
    void start() { System.out.println("Bike starts with kick"); }
}
public class VehicleTest {
    public static void main(String[] args) {
        Vehicle v1 = new Car(), v2 = new Bike();
        v1.start(); v2.start();
    }
}
```

6. Abstract BankAccount

```
abstract class BankAccount {
    abstract void accountType();
}
class SavingsAccount extends BankAccount {
    void accountType() { System.out.println("Savings Account"); }
}
class CurrentAccount extends BankAccount {
    void accountType() { System.out.println("Current Account"); }
}
public class BankTest {
    public static void main(String[] args) {
        BankAccount a1 = new SavingsAccount(), a2 = new CurrentAccount();
        a1.accountType(); a2.accountType();
    }
}
```

7. Single Inheritance

Enterprise Java Lab - Assignment 4 Solutions

```
class Person {
    String name; int age;
    Person(String n, int a) { name = n; age = a; }
}
class Employee extends Person {
    int employeeId;
    Employee(String n, int a, int id) {
        super(n, a); employeeId = id;
    }
    void display() {
        System.out.println("Name: " + name + ", Age: " + age + ", ID: " + employeeId);
    }
}
public class TestPerson {
    public static void main(String[] args) {
        Employee e = new Employee("John", 30, 101);
        e.display();
    }
}
```

8. Multilevel Inheritance

```
class Animal {
    void eat() { System.out.println("Eating"); }
}
class Mammal extends Animal {
    void walk() { System.out.println("Walking"); }
}
class Dog extends Mammal {
    void bark() { System.out.println("Barking"); }
}
public class InheritDemo {
    public static void main(String[] args) {
        Dog d = new Dog();
        d.eat(); d.walk(); d.bark();
    }
}
```

9. Dynamic Dispatch - Shape

```
class Shape {
    void draw() { System.out.println("Drawing Shape"); }
}
class Circle extends Shape {
    void draw() { System.out.println("Drawing Circle"); }
}
class Rectangle extends Shape {
    void draw() { System.out.println("Drawing Rectangle"); }
}
```

Enterprise Java Lab - Assignment 4 Solutions

```
public class DrawTest {
    public static void main(String[] args) {
        Shape s = new Circle();
        s.draw();
        s = new Rectangle();
        s.draw();
    }
}
```

10. Dynamic Dispatch - Payment

```
class Payment {
    void processPayment() { System.out.println("Processing Payment"); }
}
class CreditCardPayment extends Payment {
    void processPayment() { System.out.println("Credit Card Payment"); }
}
class PayPalPayment extends Payment {
    void processPayment() { System.out.println("PayPal Payment"); }
}
public class PaymentTest {
    public static void main(String[] args) {
        Payment p = new CreditCardPayment();
        p.processPayment();
        p = new PayPalPayment();
        p.processPayment();
    }
}
```

11. Runtime Polymorphism - Animal Sounds

```
class Animal {
    void makeSound() { System.out.println("Animal sound"); }
}
class Dog extends Animal {
    void makeSound() { System.out.println("Dog barks"); }
}
class Cat extends Animal {
    void makeSound() { System.out.println("Cat meows"); }
}
public class SoundTest {
    public static void main(String[] args) {
        Animal a = new Dog();
        a.makeSound();
        a = new Cat();
        a.makeSound();
    }
}
```

Enterprise Java Lab - Assignment 4 Solutions

12. Loan Interest (Polymorphism)

```
class Loan {
    double getInterestRate() { return 0; }
}
class HomeLoan extends Loan {
    double getInterestRate() { return 6.5; }
}
class CarLoan extends Loan {
    double getInterestRate() { return 8.0; }
}
public class LoanTest {
    public static void main(String[] args) {
        Loan l = new HomeLoan();
        System.out.println("Home Loan Interest: " + l.getInterestRate());
        l = new CarLoan();
        System.out.println("Car Loan Interest: " + l.getInterestRate());
    }
}
```

13. super in Method

```
class Animal {
    void makeSound() { System.out.println("Animal sound"); }
}
class Dog extends Animal {
    void makeSound() {
        super.makeSound();
        System.out.println("Dog barks");
    }
}
public class SuperTest {
    public static void main(String[] args) {
        Dog d = new Dog();
        d.makeSound();
    }
}
```

14. super.name Usage

```
class Person {
    String name = "Parent";
}
class Employee extends Person {
    String name = "Child";
    void showNames() {
        System.out.println("Parent Name: " + super.name);
        System.out.println("Child Name: " + name);
    }
}
public class NameTest {
    public static void main(String[] args) {
        new Employee().showNames();
    }
}
```

15. Constructor Chaining with super

```
class Employee {
    String name; int id;
    Employee(String name, int id) {
        this.name = name;
        this.id = id;
    }
}
class Cashier extends Employee {
    int registerNumber;
    Cashier(String name, int id, int regNum) {
        super(name, id);
        registerNumber = regNum;
    }
    void display() {
            System.out.println("Name: " + name + ", ID: " + id + ", Register: " +
registerNumber);
    }
}
public class CashierTest {
    public static void main(String[] args) {
        new Cashier("Ravi", 101, 12).display();
    }
}
```

16. Static Variable to Count Objects

```
class Counter {
    static int count = 0;
    Counter() { count++; }
    static void showCount() {
        System.out.println("Objects created: " + count);
    }
}
public class StaticCount {
    public static void main(String[] args) {
        new Counter(); new Counter(); new Counter();
        Counter.showCount();
    }
}
```

Enterprise Java Lab - Assignment 4 Solutions

17. Static Method to Square

```
class MathUtil {
    static int square(int x) { return x * x; }
}
public class SquareTest {
    public static void main(String[] args) {
        System.out.println("Square: " + MathUtil.square(5));
    }
}
```

18. Static Block

```
public class StaticBlock {
    static {
        System.out.println("Static block executed");
    }
    public static void main(String[] args) {
        System.out.println("Main method");
    }
}
```

19. Static Method Accessing Non-static Variable

```
class Example {
    static int a = 10;
    int b = 20;
    static void show() {
        Example e = new Example();
        System.out.println("a: " + a + ", b: " + e.b);
    }
}
public class StaticAccess {
    public static void main(String[] args) {
        Example.show();
    }
}
```

20. Inheriting Static Method

```
class Parent {
    static void show() { System.out.println("Static from Parent"); }
}
class Child extends Parent {}
public class StaticInherit {
    public static void main(String[] args) {
        Parent.show();
        Child.show();
    }
}
```

21. Static Nested Class

```
class Outer {
    static class Nested {
        void display() {
            System.out.println("Static Nested Class");
        }
    }
}
public class NestedTest {
    public static void main(String[] args) {
        Outer.Nested n = new Outer.Nested();
        n.display();
    }
}
```