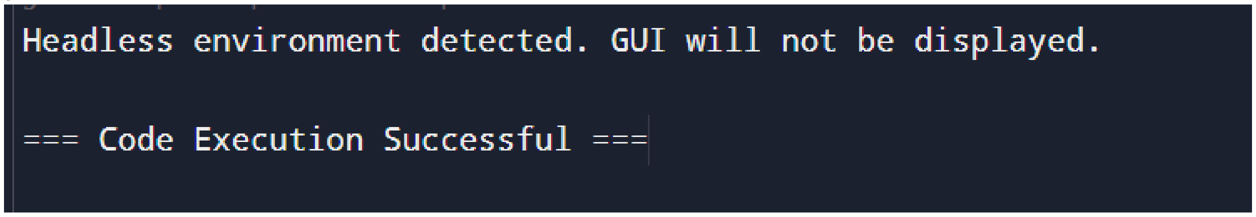
Java Programming

Section 3.1 practice

1. JavaBank Application

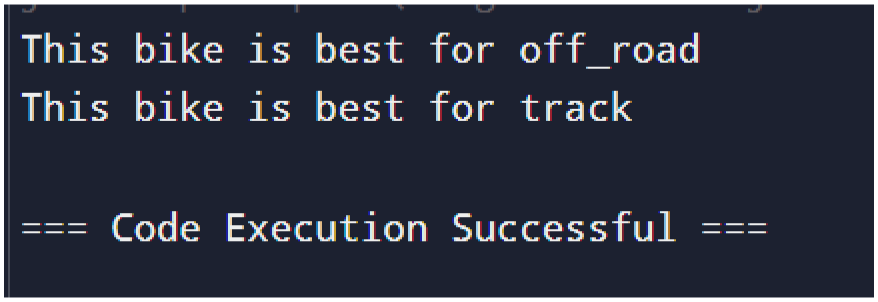
```
import javax.swing.*;
import java.awt.*;

public class JavaBank {
    // Combo box to display the account types
    private JComboBox<AccountType> accountTypes;
    private AccountType actType = AccountType.SAVINGS;

    // Constants
    public static final int MAXACCOUNTS = 10;
    private AbstractBankAccount[] myAccounts = new AbstractBankAccount[MAXACCOUNTS];
    private int numberOfAccounts = 0;

    // GUI components
    private JPanel inputDetailJPanel;
    private JTextArea displayJTextArea;

    public JavaBank() {
        if (!GraphicsEnvironment.isHeadless()) {
            createUserInterface();
        } else {
            System.out.println("Headless environment detected. GUI will not be displayed.");
        }
    }

    private void createUserInterface() {
        // Setup main frame
        JFrame frame = new JFrame("JavaBank");
        frame.setDefaultCloseOperation(JFrame.EXIT_ON_CLOSE);
        frame.setLayout(null);

        // Setup inputDetailJPanel
        inputDetailJPanel = new JPanel();
        inputDetailJPanel.setLayout(null);
        inputDetailJPanel.setBounds(16, 16, 208, 280);
        frame.add(inputDetailJPanel);
        // Set up accountTypes combo box
        accountTypes = new JComboBox<>(AccountType.values());
        accountTypes.setBounds(16, 238, 176, 24);
        inputDetailJPanel.add(accountTypes);
        accountTypes.addActionListener(
            event -> actType = (AccountType) accountTypes.getSelectedItem()
        );

        // Setup displayJTextArea
        displayJTextArea = new JTextArea();
        displayJTextArea.setBounds(240, 16, 400, 245);
        frame.add(displayJTextArea);

        // Set the window size
        frame.setSize(670, 340);
        frame.setVisible(true);
    }

    public static void main(String[] args) {
        new JavaBank();
    }
}
// Enum for Account Types
enum AccountType {
    SAVINGS, CREDIT
}

// Abstract class for bank accounts
abstract class AbstractBankAccount {
    // Abstract methods and common properties for bank accounts
}

// Savings Account Class
class SavingsAccount extends AbstractBankAccount {
    // Implementation for savings account
}

// Credit Account Class
class CreditAccount extends AbstractBankAccount {
    // Implementation for credit account
}
```



```
Headless environment detected. GUI will not be displayed.

=== Code Execution Successful ===
```

2. Bike Project

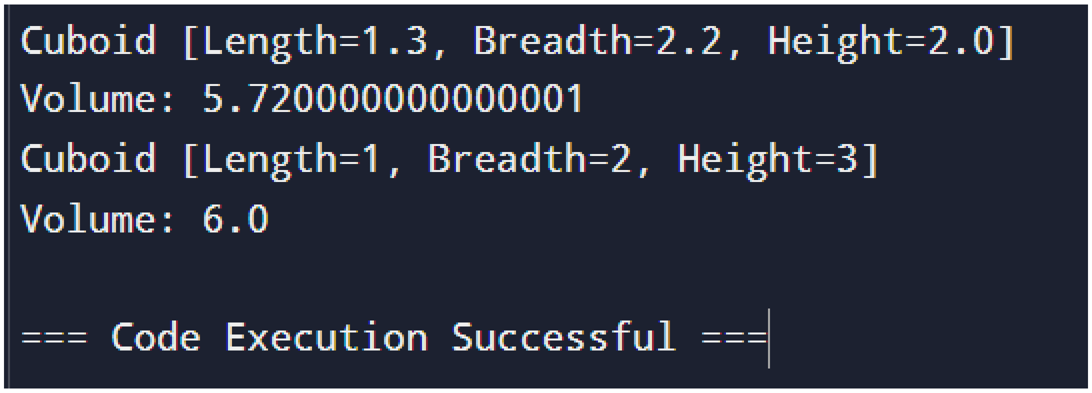
```
// Enum for Bike Uses
enum BikeUses {
    OFF_ROAD, TRACK, ROAD, DOWNHILL, TRAIL
}

// MountainParts Interface
interface MountainParts {
    // Using Enum for terrain
    BikeUses terrain = BikeUses.OFF_ROAD;

    // Other methods...
}

// RoadParts Interface
interface RoadParts {
    // Using Enum for terrain
    BikeUses terrain = BikeUses.TRACK;

    // Other methods...
}

// MountainBike Class
class MountainBike implements MountainParts {
    // Fields and methods...

    @Override
    public String toString() {
        return "This bike is best for " + terrain.toString().toLowerCase();
    }
}

// RoadBike Class
class RoadBike implements RoadParts {
    // Fields and methods...

    @Override
    public String toString() {
        return "This bike is best for " + terrain.toString().toLowerCase();
    }
}

public class BikeProject {
    public static void main(String[] args) {
        MountainBike mountainBike = new MountainBike();
        RoadBike roadBike = new RoadBike();
        System.out.println(mountainBike);
        System.out.println(roadBike);
    }
}
```



```
This bike is best for off_road
This bike is best for track

=== Code Execution Successful ===
```

3. Generic Shapes Project

```
// Save this file as CuboidProject.java

public class CuboidProject {

    public static void main(String[] args) {
        // Test the Cuboid with Double dimensions
        Cuboid<Double> c1 = new Cuboid<>();
        c1.setLength(1.3);
        c1.setBreadth(2.2);
        c1.setHeight(2.0);
        System.out.println(c1);
        System.out.println("Volume: " + c1.getVolume());

        // Test the Cuboid with Integer dimensions
        Cuboid<Integer> c2 = new Cuboid<>();
        c2.setLength(1);
        c2.setBreadth(2);
        c2.setHeight(3);
        System.out.println(c2);
        System.out.println("Volume: " + c2.getVolume());
    }

    // Cuboid Class
    static class Cuboid<T extends Number> {
        private T length, breadth, height;

        public void setLength(T length) {
            this.length = length;
        }

        public void setBreadth(T breadth) {
            this.breadth = breadth;
        }
        public void setHeight(T height) {
            this.height = height;
        }

        public T getLength() {
            return length;
        }

        public T getBreadth() {
            return breadth;
        }

        public T getHeight() {
            return height;
        }

        public double getVolume() {
            return length.doubleValue() * breadth.doubleValue() * height.doubleValue();
        }

        @Override
        public String toString() {
            return "Cuboid [Length=" + length + ", Breadth=" + breadth + ", Height=" + height + "]";
        }
    }
}
```



```
Cuboid [Length=1.3, Breadth=2.2, Height=2.0]
Volume: 5.720000000000001
Cuboid [Length=1, Breadth=2, Height=3]
Volume: 6.0

=== Code Execution Successful ===
```