Java Programming 3-1: Generics Practice

Solution

Vocabulary Definitions

1.

This is a special type of class that associates one or more non-specified Java types.

2.

- **Generic Class**: A class that allows the specification of types at runtime.

3.

A type interface diamond is what 2 characters?

4.

- **<>**: The type interface diamond refers to the angle brackets used in generics, which simplify type declarations.

5.

A datatype that contains a fixed set of constants.

6.

- **Enum**: An enumerated type in Java, which represents a fixed set of constants.

JavaBank Application Update

1. Updating the JavaBank Application

a. **Add JComboBox for Account Types**

Update `JavaBank.java` to include the `JComboBox` for account types and a field to store the selected type:

```
private JComboBox<AccountType> accountTypes;
private AccountType actType = AccountType.SAVINGS;
```

b. **Setup JComboBox and ActionListener**

Add this code snippet to set up the `JComboBox` and handle user selections:

```
// set up accountTypes combo box
accountTypes = new JComboBox<AccountType>(AccountType.values());
accountTypes.setBounds(16, 238, 176, 24);
inputDetailJPanel.add(accountTypes);
accountTypes.addActionListener(new ActionListener() {
   public void actionPerformed(ActionEvent event) {
      actType = (AccountType) accountTypes.getSelectedItem();
   }
});
```

c. **Update the Window Size**

To fix the issue with the combo box display:

```
// Update window size
setSize(670, 340); // Increase height
```

d. **Update Panel Size**

Adjust the panel bounds to accommodate the new height:

```
inputDetailJPanel.setBounds(16, 16, 208, 280); // Increase height by 30 pixels
```

e. **Update Text Area Size**

Adjust the text area bounds:

```
displayJTextArea.setBounds(16, 280, 208, 245); // Increase height
```

f. **Update Array Declaration**

Change the array type to `AbstractBankAccount`:

```
AbstractBankAccount[] myAccounts = new AbstractBankAccount[MAXACCOUNTS];
```

g. **Update Account Type Check**

Update the `createAccountJButtonActionPerformed` method to check the account type before adding:

```
if (actType == AccountType.SAVINGS) {
    myAccounts[count] = new SavingsAccount(...);
} else if (actType == AccountType.CHECKING) {
    myAccounts[count] = new CheckingAccount(...);
}
```

Bike Project Updates

2. Enum and Interface Updates

a. **Create Enum Class**

Define the `BikeUses` enum:

```
public enum BikeUses {
    OFF_ROAD,
    TRACK,
    ROAD,
    DOWNHILL,
    TRAIL
}
```

b. **Update Interfaces**

-

MountainParts Interface

-

```
public interface MountainParts {
    BikeUses terrain = BikeUses.OFF_ROAD;
}
```

RoadParts Interface

```
public interface RoadParts {
    BikeUses terrain = BikeUses.TRACK;
}
```

c. **Update toString() Methods**

Modify the `toString()` methods to include terrain information:

java

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

d. **Test Program**

Run and test the application to ensure it displays the correct information.

Generic Shapes Project

3. Creating and Using Generics

a. **Create Generic Class** `Cuboid`

Define the `Cuboid` class:

```java
public class Cuboid<T extends Number> {
    private T length;
    private T breadth;
    private T height;

    public Cuboid(T length, T breadth, T height) {
        this.length = length;
        this.breadth = breadth;
        this.height = height;
    }

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

    public T getBreadth() { return breadth; }
    public void setBreadth(T breadth) { this.breadth = breadth; }

    public T getHeight() { return height; }
    public void setHeight(T height) { this.height = height; }

    @Override
    public String toString() {
        return "Length: " + length + ", Breadth: " + breadth + ", Height: " + 
height;
    }

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

b. **Driver Class for Double Cuboid**

Instantiate `Cuboid<Double>` and display its volume:

```
public class CuboidDriver {
    public static void main(String[] args) {
        Cuboid<Double> doubleCuboid = new Cuboid<>(1.3, 2.2, 2.0);
        System.out.println(doubleCuboid);
        System.out.println("Volume: " + doubleCuboid.getVolume());
    }
}
```

c. **Driver Class for Integer Cuboid**

Instantiate `Cuboid<Integer>` and display its volume:

```
public class CuboidDriverInteger {
    public static void main(String[] args) {
        Cuboid<Integer> integerCuboid = new Cuboid<>(1, 2, 3);
        System.out.println(integerCuboid);
        System.out.println("Volume: " + integerCuboid.getVolume());
    }
}
```