# Java Programming 2-2: Java Class Design - Interfaces Practice Activities

## Vocabulary Section

1. **A specialized method that creates an instance of a class**: **Constructor**.
2. **A keyword that qualifies a variable as a constant and prevents a method from being overridden in a subclass**: `final`.
3. **A class that can't be overridden by a subclass, in fact, it can't be subclassed**: `final` **class**.
4. **Defines constants and methods without implementation**: **Interface**.

## JavaBank Update: Implement New Company Color

1. **Update Company Color**:

    1. To set the new color across all GUI elements, you'll likely need to update the colorsettings in the GUI code. For instance, if you're using Swing:

```
Color companyColor = new Color(173, 216, 230); // Light blue color
someComponent.setBackground(companyColor); // Apply to GUI components
```

    1. You need to locate all GUI elements and update their background colors to use thisnew color.

## Creating and Implementing Interfaces in the Bike Project

### BikeParts Interface:

1. Define the `BikeParts` interface as instructed:

Code:

```
package bikeproject;



public interface BikeParts {

   // Constant declaration

   public final String MAKE = "Oracle Bikes";



   // Required methods

   public String getHandleBars();

   public void setHandleBars(String newValue);
```

```
            public String getTyres();

            public void setTyres(String newValue);

            public String getSeatType();

            public void setSeatType(String newValue);

        }
```

**3.MountainParts Interface**:

- Code:

```
package bikeproject;


public interface MountainParts {

    // Constant declaration

    public final String TERRAIN = "off_road";


    // Required methods

    public String getSuspension();

    public void setSuspension(String newValue);

    public String getType();

    public void setType(String newValue);

}
```

5. RoadParts Interface:

```
package bikeproject;


public interface RoadParts {

    // Constant declaration

    public final String TERRAIN = ″track_racing″;
```

```
        // Required methods
        public String getTyreWidth();
        public void setTyreWidth(String newValue);
        public String getPostHeight();
        public void setPostHeight(String newValue);
    }
```

## Implementing Interfaces in Classes

1. **Implement `BikeParts` in `Bike` Class**:

```
package bikeproject;


public class Bike implements BikeParts {
    private String handleBars;
    private String tyres;
    private String seatType;

    // Implement required methods
    @Override
    public String getHandleBars() {
        return handleBars;
    }


    public void setHandleBars(String newValue) {
        this.handleBars = newValue;
    }

    public String getTyres() {
        return tyres;
    }

    @Override
    public void setTyres(String newValue) {
        this.tyres = newValue;
    }

    @Override
    public String getSeatType() {
        return seatType;
    }

    @Override
    public void setSeatType(String newValue) {
        this.seatType = newValue;
    }

    // Other existing code
}
```

Implement MountainParts in MountainBike Class:

```
package bikeproject;


public class MountainBike extends Bike implements MountainParts
{

    private String suspension;

    private String type;


    // Implement required methods

    @Override

    public String getSuspension() {

        return suspension;

    }


    @Override

    public void setSuspension(String newValue) {

        this.suspension = newValue;

    }


    @Override

    public String getType() {

        return type;

    }
    @Override
    public void setType(String newValue) {
        this.type = newValue;
    }

    // Other existing code
}
```

Implement MountainParts in MountainBike Class:

```
package bikeproject;

public class MountainBike extends Bike implements MountainParts {
    private String suspension;
    private String type;

    // Implement required methods
    @Override
    public String getSuspension() {
        return suspension;
    }

    @Override
    public void setSuspension(String newValue) {
        this.suspension = newValue;
        }

        @Override
        public String getType() {
            return type;
        }

        @Override
        public void setType(String newValue) {
            this.type = newValue;
        }

        // Other existing code
    }
```

7. Implement RoadParts in RoadBike Class:

```
package bikeproject;

public class RoadBike extends Bike implements RoadParts {
    private String tyreWidth;
    private String postHeight;

    // Implement required methods
    @Override
    public String getTyreWidth() {
        return tyreWidth;

    }



    @Override

    public void setTyreWidth(String newValue) {

        this.tyreWidth = newValue;

    }



    @Override

    public String getPostHeight() {

        return postHeight;

    }



    @Override

    public void setPostHeight(String newValue) {

        this.postHeight = newValue;

    }



    // Other existing code

}
```

- **Run and Test the Program**:
  - • Ensure that the program behaves as expected after the changes. It should work just as it did before.
- **Update the Height of the Post for `bike1`**:

- • At the bottom of your driver class (the main method), update the `postHeight` value for `bike1`:

```
bike1.setPostHeight("20"); // Set the post height to 20
```

## 10. Display the Values of `bike1`:

- Print out the details of `bike1` to confirm that the `postHeight` has been updated

```
System.out.println("Bike1 Post Height: " + bike1.getPostHeight());
```

## 11. Run and Test Your Program Again:

- Verify that the `postHeight` is correctly updated and displayed as 20 instead of 22.