1. Package Balance with Class Account

Create a package named Balance containing a class Account with a method Display_Balance.

Balance/Account.java:

```java
package Balance;

public class Account {
    private double balance;

    public Account(double balance) {
        this.balance = balance;
    }

    public void Display_Balance() {
        System.out.println("Account Balance: " + balance);
    }
}
```

Main.java:

```java
import Balance.Account;

public class Main {
    public static void main(String[] args) {
        Account account = new Account(1000.50);
        account.Display_Balance();
    }
}
```

Output:

```
Account Balance: 1000.5
```

---

2. Package p with Class A and Package Q with Class B

Create a package p containing a class A with methods having different access modifiers. Then, create a class B in package Q to demonstrate access.

p/A.java:

```java
package p;

public class A {
    public void publicMethod() {
        System.out.println("Public Method");
    }
    protected void protectedMethod() {
        System.out.println("Protected Method");
    }

    void defaultMethod() {
        System.out.println("Default Method");
    }

    private void privateMethod() {
        System.out.println("Private Method");
    }
}
```

Q/B.java:

```java
package Q;
import p.A;

public class B {
    public static void main(String[] args) {
        A obj = new A();

        // Access public method
        obj.publicMethod();

        // Access protected method (only if B extends A)
        // obj.protectedMethod(); // This will cause a compilation error

        // Access default method (only within the same package)
        // obj.defaultMethod(); // This will cause a compilation error

        // Access private method (not allowed)
        // obj.privateMethod(); // This will cause a compilation error
    }
}
```

Output:

```
Public Method
```

---

3. Use of `final` Keyword with Variable and Method

Create a class `MathConstants` with a `final` variable `PI` and a `final` method `displayPI()`. Extend it in another class `Circle`.

MathConstants.java:

```
public class MathConstants {
    public static final double PI = 3.14159;

    public final void displayPI() {
        System.out.println("PI: " + PI);
    }
}
```

Circle.java:

```
public class Circle extends MathConstants {
    public void calculateArea(double radius) {
        double area = PI * radius * radius;
        System.out.println("Area of Circle: " + area);
    }

    // Uncommenting this will cause a compilation error
    // @Override
    // public void displayPI() {
    //     System.out.println("Overridden PI: " + PI);
    // }
}
```

Main.java:

```
public class Main {
    public static void main(String[] args) {
        Circle circle = new Circle();
        circle.displayPI();
        circle.calculateArea(5.0);

        // Uncommenting this will cause a compilation error
        // MathConstants.PI = 3.14;
    }
}
```

Output:

```
PI: 3.14159
Area of Circle: 78.53975
```

---

4. Use of `final` Keyword with Class

Create a `final` class `Logger` and attempt to extend it.

Logger.java:

```java
public final class Logger {
    public void logMessage(String message) {
        System.out.println("Log: " + message);
    }
}
```

ExtendedLogger.java:

```java
// Uncommenting this will cause a compilation error
// public class ExtendedLogger extends Logger {
//     public void logMessage(String message) {
//         System.out.println("Extended Log: " + message);
//     }
// }
```

Main.java:

```java
public class Main {
    public static void main(String[] args) {
        Logger logger = new Logger();
        logger.logMessage("This is a log message.");
    }
}
```

Output:

```
Log: This is a log message.
```