Interfaces and multiple interface implementation:

In Java, a class can implement multiple interfaces, which allows it to inherit and define behavior from multiple sources. When a class implements multiple interfaces, it needs to provide an implementation for all the methods defined in those interfaces. Let's go through some details about multiple interface implementation in Java:

1. **Syntax:**

javaCopy code
```
public class MyClass implements Interface1, Interface2, Interface3 {
   // Class implementation
}
```

2. **Method Overriding:** If multiple interfaces have a method with the same signature, the implementing class must provide an implementation for that method. It should adhere to the method signature specified in each interface. The class cannot differentiate between the interfaces, so it provides a single implementation for the method.
3. **Conflicting Default Methods:** A conflict arises when multiple interfaces provide default methods with the same name and signature. In such cases, the implementing class must explicitly override the conflicting default method and provide its own implementation. Otherwise, a compilation error occurs. For example:

javaCopy code
```
interface Interface1 {
   default void foo() {
      System.out.println("Interface1");
   }
}

interface Interface2 {
   default void foo() {
      System.out.println("Interface2");
   }
}

class MyClass implements Interface1, Interface2 {
   @Override
   public void foo() {
      Interface1.super.foo(); // Specify which default method to use
   }
}
```

4. **Interface Inheritance:** If two interfaces have a parent-child relationship, i.e., one extends the other, implementing the child interface automatically implies implementing the parent interface. The implementing class must provide implementations for all methods from both interfaces. For example:

```java
javaCopy code
interface ParentInterface {
   void parentMethod();
}

interface ChildInterface extends ParentInterface {
   void childMethod();
}

class MyClass implements ChildInterface {
   @Override
   public void parentMethod() {
      // Implementation
   }

   @Override
   public void childMethod() {
      // Implementation
   }
}
```

5. **Multiple Inheritance:** Java doesn't support multiple inheritance of classes, i.e., a class cannot extend multiple classes. However, it does allow multiple inheritance of interfaces, where a class can implement multiple interfaces. This feature enables a class to inherit behavior from multiple sources.

It's important to note that implementing multiple interfaces can lead to complex relationships and may require careful design to avoid conflicts and maintain code clarity.

Class Example:

Employee interface:

```java
package util.multi;

public interface Employee extends Salaried{
   String getName();
   void commonMethod();
}
```

Identity interface :

```java
package util.multi;

public interface Identity {
   String getId();
}
```

Salaried interface:

```java
package util.multi;

public interface Salaried{
   float getSalary();
   void commonMethod();
}
```

Person class implementation:

```java
package util.multi;

public class Person implements Employee, Identity{
   String name;
   float salary;
   String id;

   public Person(String name, float salary, String id) {
      this.name = name;
      this.salary = salary;
      this.id = id;
   }

   @Override
   public String getName() {
      return this.name;
   }

   @Override
   public void commonMethod() {
      System.out.println("Execution common method");
   }

   @Override
   public float getSalary() {
      return this.salary;
   }

   @Override
   public String getId() {
      return this.id;
   }
}
```

Person class runner:

```java
package util.multi;

public class PersonDemo {
   public static void main(String[] args) {
      Person p1 = new Person("John", 100f, "1");
      Employee p2 = new Person("John1", 200f, "2");
      Salaried p3 = new Person("John3", 300f, "3");
      Identity p4 = new Person("John4", 300f, "4");


      System.out.println("Name is : "+p1.getName());
      System.out.println("Salary is : "+p1.getSalary());
      p1.commonMethod();

      System.out.println("Name is : "+p2.getName());
      System.out.println("Salary is : "+p2.getSalary());
      p2.commonMethod();

      //Salaried reference variable cannot access getName method
      //System.out.println("Name is : "+p3.getName());
      System.out.println("Salary is : "+p3.getSalary());
      p3.commonMethod();

      //Identity reference variable cannot access getSalary, getName
      // and common method as those are not declared in it
      //System.out.println("Name is : "+p4.getName());
      //System.out.println("Salary is : "+p4.getSalary());
      System.out.println("Salary is : "+p4.getId());
      //p4.commonMethod();
   }
}
```

Assignment question:
Create interface called Shape which has method getName
Create interface Drawable which has method draw
        Drawable should extend interface called Calculator
Calculator has method calculatePerimeter and calculateArea method

Create class Circle, Rectangle, and Square which implement both Shape and Drawable interfaces
Add required instance variables to Circle, Rectangle and Square class to calculate perimeter and area

Create object of circle and square and print call draw method and calculatePerimeter and Area method

hint : perimeter of circle is = 2 * 3.14 * radius
        area of circle = 3.14 * radius * radius
        peremeter of rectangle = 2 * length + 2 * breadth

area of rectangle = length * breadth
area of square = side * side
peremeter of square = 4 * sides
area of square = side * side