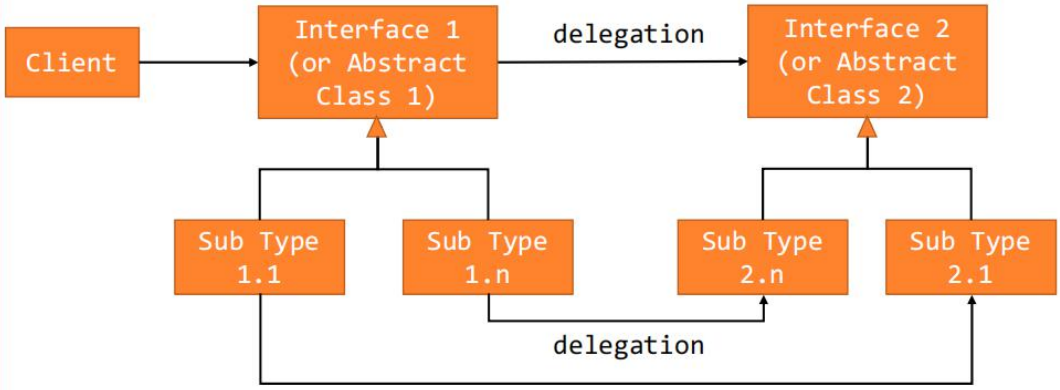
1. 写接口时不用写大括号，只要写方法签名即可（权限，类型，参数）
2. 判断相等时最好将方法传进来的参数放在前面，如 label.equals(tmp.getLabel())
3. 构造器方法 public MultiIntervalSet(){}
4. 检查方法 checkRep private void checkRep() {}
5. 写接口一定要写 empty(),否则实现的类会报错
6. Composite Reuse Principle (CRP) 复合重用原则：类通过“组合”实现多态和复用（通过引入其他类的实例来实现功能）而不是通过基类或父类。



(1) 写接口：定义抽象行为

```
interface Flyable {
public void fly();
}
interface Quackable {
public void quack();
}
```

(2) 接口的具体实现：可以有多种方式，在具体类中实现具体行为

```
class FlyWithWings implements Flyable {
 @Override
 public void fly() {
 System.out.println("fly with wings");
 }
}
class Quack implements Quackable {
 @Override
 public void quack() {
 System.out.println("quack like duck");
 }
}
```

(3) 接口的组合：定义了行为的组合

```
interface Ducklike extends Flyable, Quackable {}
```

(4) 从组合接口中派生具体类

```
public class Duck implements Ducklike {}
```

(5) 委派：设置 Delegation 对象实例，通过 Delegation 实现具体行为

```
Flyable flyBehavior;
 Quackable quackBehavior;
 void setFlyBehavior(Flyable f) {
 this.flyBehavior = f;
 }
 void setQuackBehavior(Quackable q) {
 this.quackBehavior = q;
 }
 @Override
 public void fly() {
 this.flyBehavior.fly();
 }
 @Override
 public void quack() {
 this.quackBehavior.quack();
 }
```