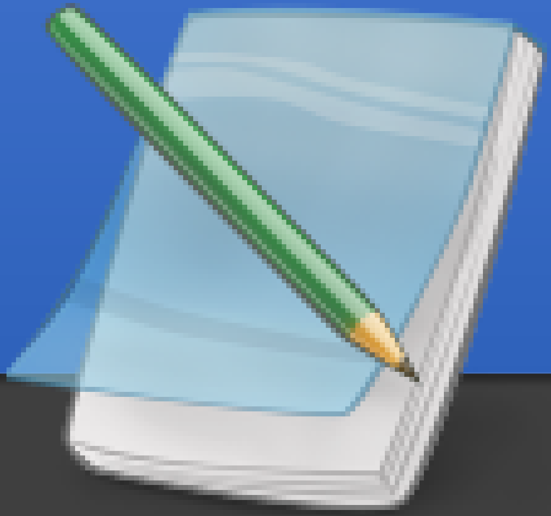
Meow!
Woof!
Quack!
Animal
Animal
Animal
Polymorphism
speak();
by Sinipull for codecall.net

Lab 6
Interface & Polymorphism

KEYNOTE

- Abstract Class
- Interface
- Abstract Class vs. Interface
- Polymorphism



1. Abstract Class

- A class in which the programmer never intends to instantiate any objects.
- The purpose is to provide an appropriate superclass from which other classes may inherit interface and force prototype standards to be followed.

1. Abstract Class

Keyword abstract

```
public abstract class Animal {
 private String name;

 // constructor
 public Animal(String name) {
  this.name = name;
 }

 // get method
 public String getName() {
  return name;
 }

 // abstract method has no implement
 public abstract void speak();
}
```

at least one abstract method

```
class Dog extends Animal {
 public Dog(String name) {
  super(name);
 }

 @Override
 public void speak() {
  System.out.println("Bow bow");
 }
}
```

Abstract method(s) must be implemented

declare references to abstract superclasses

```
public static void main(String[] args) {
 Animal ani1 = new Dog("Hachiko");
 ani1.speak();
}
```

2. Interface

- Interfaces
  - Interfaces are similar to abstract classes but all methods are public, abstract and all properties are public, static, final.
  - Java does not allow multiple inheritance for classes. An interface is used to tie elements of several classes together.
  - Note:
    - Interfaces can be inherited (*i.e. you can have a sub-interface*). As with classes the *extends* keyword is used for inheritance.
    - Java allow a class to inherit from a superclass and implement more than one interface.

2. Interface

Keyword interface

Use the interface

```
interface ISpeak{
 public void speak();
}

class Cat implements ISpeak{
 @Override
 public void speak() {
   System.out.println(“Meow meow");
 }
}
```

3. Abstract Class vs. Interface

Abstract Class	**Interface**
Abstract classes may contain only abstract declarations **and/or** concrete implementations.	Interfaces are rules, all methods must be **public**.
Abstract declarations are like rules to be followed and concrete implementations are like guidelines.	Interfaces give the idea what is to be done but **not how it will be done**.
Has “Is a” relationship	Unrelated but share the “abilities”

4. Polymorphism

- Definition:
  - Subclasses of a class can define their own unique behaviors and yet share some of the same functionality of the parent class
- How To Do:
  - In polymorphism , the method in the subclass that has the same signature as the one in the superclass (overriding).
  - The program must determine which version of an overridden method to call at runtime by the type of the actual object, NOT the type of the object reference.

PRACTICE

1. Exercise 1

- Objective:
  - Implement with Abstract Class and Interface
  - Practice about Polymorphism

To Do:

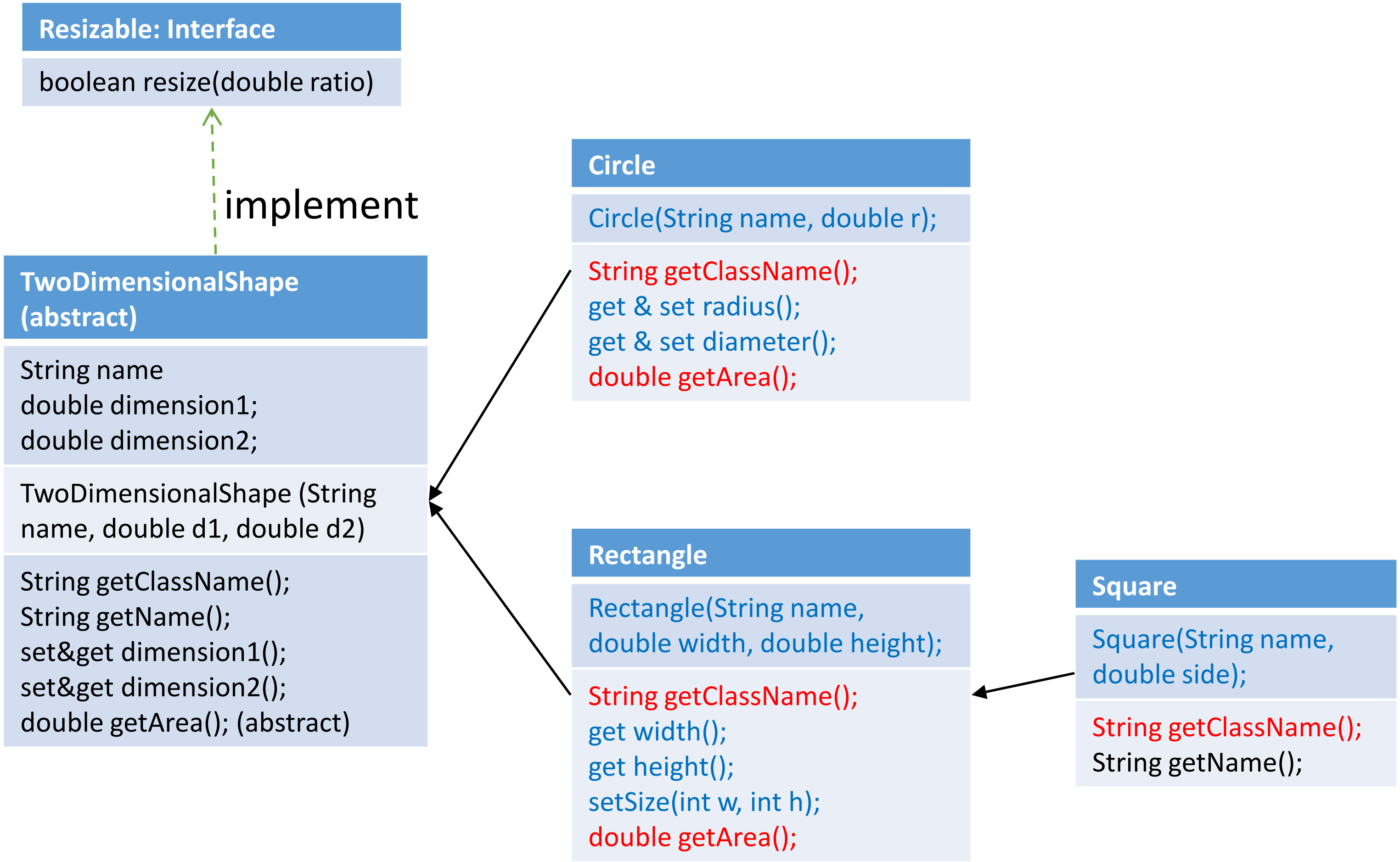
1. Download the attachment file in E-Class. We have the code implement hierarchies classes: Circle, Rectangle, Square, TwoDimensionalShape.
2. Implement an interface Sizable, which has method "*void resize(double ratio)".* (The area of the shape will be change *ratio* times)
   - *If the shape is Rectangle -> width = width * ratio;*
   - *If the shape is Circle/Square -> radius/side = radius/side * sqrt(ratio);*
3. Create a TwoDimensionalShape reference array which has mix of Circle, Rectangle, Square object. Print the size of object before and after resize.

1. Exercise 1

4. Implement method to compare the area of any two 2D objects (such as: Circle, Rectangle, Square). It return 0 if equal, -1 if smaller, +1 if larger.
   Example: *cir1.compareTo(rec1), sq1.compareTo(rec1) ...*

- Hint:
  the 2DShape class have to implement interface Comparable<T>

1. Exercise 1



Resizable: Interface

boolean resize(double ratio)

implement

TwoDimensionalShape (abstract)

String name
double dimension1;
double dimension2;

TwoDimensionalShape (String name, double d1, double d2)

String getClassName();
String getName();
set&get dimension1();
set&get dimension2();
double getArea(); (abstract)

Circle

Circle(String name, double r);

String getClassName();
get & set radius();
get & set diameter();
double getArea();

Rectangle

Rectangle(String name, double width, double height);

String getClassName();
get width();
get height();
setSize(int w, int h);
double getArea();

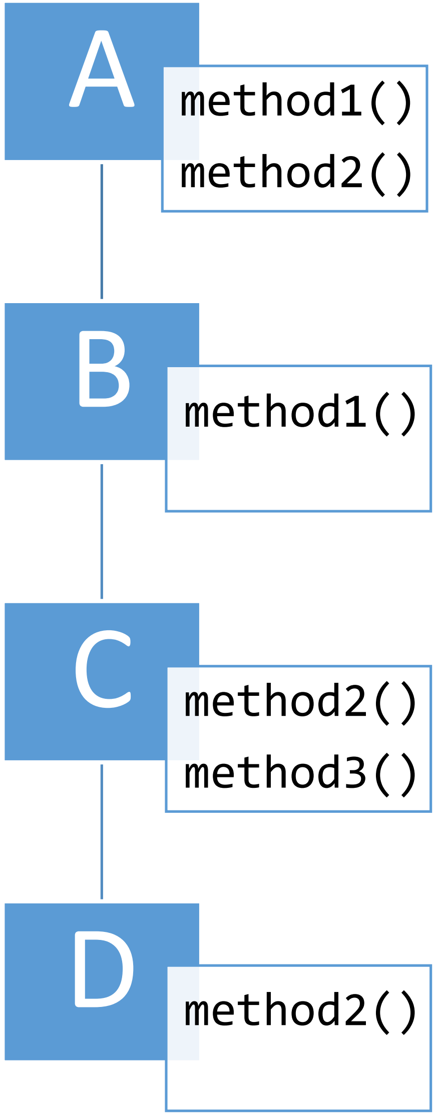
Square

Square(String name, double side);

String getClassName();
String getName();

2. Exercise 2

```
public class A {
  public void method1() {
     System.out.println("A1");
  }
  public void method2() {
     method1();
     System.out.println("A2");
  }
}
public class B extends A {
  public void method1() {
     System.out.println("B1");
     super.method1();
  }
}
public class C extends B {
  public void method2() {
     System.out.println("C2");
  }
  public void method3() {
     super.method1();
     System.out.println("C3");
  }
}
public class D extends C {
  public void method2() {
     super.method2();
     System.out.println("D2");
  }
}
```



2. Exercise 2

What is the output of test program?

```java
public class Test {
  public static void main(String[] args){
      A var1 = new A();
      A var2 = new B();
      B var3 = new D();
      C var4 = new C();
      D var5 = new D();
      Object var6 = new B ();

      //Methods are called here
      //Guest the output
  }
}
```

1. A1
2. B1/A1

II. Exercise 2 - Answer

1.	var1.method1();	1.	A1
2.	var2.method1();	2.	B1 \ A1
3.	var3.method1();	3.	B1 \ A1
4.	var4.method1();	4.	B1 \ A1
5.	var5.method1();	5.	B1 \ A1
6.	var6.method1();	6.	compiler error
7.	var1.method2();	7.	A1 \ A2
8.	var2.method2();	8.	B1 \ A1 \ A2
9.	var3.method2();	9.	C2 \ D2
10.	var4.method2();	10.	C2
11.	var5.method2();	11.	C2 \ D2
12.	var6.method2();	12.	compiler error
13.	var3.method3();	13.	compiler error
14.	var5.method3();	14.	B1 \ A1 \ C3
15.	((B) var1).method1();	15.	runtime error
16.	((C) var2).method2();	16.	runtime error
17.	((D) var5).method1();	17.	B1 \ A1
18.	((C) var3).method3();	18.	B1 \ A1 \ C3
19.	((D) var4).method3();	19.	runtime error
20.	((C) var6).method3();	20.	runtime error