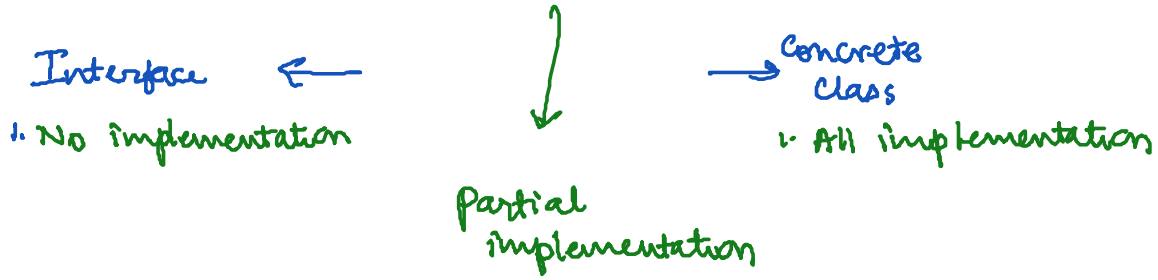
Java Abstract Class

Monday, October 25, 2021 2:11 PM

Java_Abst...
Classes_7a

Java
Abstract Class



Concept

- `interface` is a way to describe what classes should do, without specifying how they should do it.

- What if we want to also specify part of the functionality
  - One possible solution is to have a dummy class: with functions
  - Have all classes extend this dummy class

A problem with interfaces

```
public interface Shape2D {
   int getX();
   int getY();
   double getArea();
   double getPerimeter();
}
```

- Every shape will implement `getX` and `getY` the same- Cant give an implementation in an interface
- but each shape probably implements `getArea` and `getPerimeter` differently

3

A bad solution

```
public class Shape implements Shape2D {
  private int myX, myY;

  public Shape(int x, int y) {
     myX = x;  myY = y;
  }
  public int getX() { return myX; }
  public int getY() { return myY; }

  // subclasses should override these,
  please
  public double getArea() { return 0; }
  public double getPerimeter() { return 0; }
}
```

- BAD: the `Shape` class can be instantiated
- BAD: a subclass might forget to override the methods

4

Abstract classes

- **abstract class**: a hybrid between an interface and a class
  - used to define a generic parent type that can contain method declarations (like an interface) and/or method bodies (like a class)
  - like interfaces, abstract classes that cannot be instantiated (cannot use `new` to create any objects of their type)

What goes in an abstract class?

- implement common state and behavior that will be inherited by subclasses (parent class role)
- declare generic behaviors that subclasses must implement (interface role)

5

Abstract class syntax

- put `abstract` keyword on class header and on any generic (abstract) methods
  - A class can be declared `abstract` even though it has no abstract methods
  - Any class with abstract methods *must* be declared `abstract`, or it will not compile
- Variables of abstract types may be declared, but *objects* of abstract types cannot be constructed

6

Abstract class example

```
public abstract class Shape implements
  Shape2D {
  private int myX, myY;

  public Shape(int x, int y) {
     myX = x;  myY = y;
  }
  public int getX() { return myX; }
  public int getY() { return myY; }

  public abstract double getArea();
  public abstract double getPerimeter();
}
```

- `Shape` class cannot be instantiated
- all classes that extend `Shape` **must** implement `getArea` and `getPerimeter` or else must also be declared `abstract`

7

Extending an abstract class

```
public class Rectangle extends Shape {
  private int myWidth, myHeight;

  public Rectangle(int x,int y,int w, int h)
  {
    super(x, y);
    myWidth = w;   myHeight = h;
  }

  public double getArea() {
    return myWidth * myHeight;
  }
  public double getPerimeter() {
    return 2*myWidth + 2*myHeight;
  }
}

// ... example usage ...
Shape rect = new Rectangle(1, 2, 10, 5);
```

8

Interface / abstract class chart

TABLE 3.1
Comparison of Actual Classes, Abstract Classes, and Interfaces

Property	Actual Class	Abstract Class	Interface
Instances (objects) of this can be created	Yes	No	No
This can define instance variables and methods	Yes	Yes	No
This can define constants	Yes	Yes	Yes
The number of these a class can extend	0 or 1	0 or 1	0
The number of these a class can implement	0	0	Any number
This can extend another class	Yes	Yes	No
This can declare abstract methods	No	Yes	Yes
Variables of this type can be declared	Yes	Yes	Yes