目录

Day07.Java

1 回顾

- 类
  - 模板、图纸
  - 封装相关的属性数据，方法代码
- 对象
  - 实例
  - 占用独立内存空间，保存属性数据
  - 可以独立控制，执行指定方法的代码
- 引用
  - 遥控器
  - 保存对象的内存地址
  - 引用的特殊值 `null`
- 构造方法
  - 新建对象时执行
  - 不定义，会添加默认构造方法
  - 构造方法重载
  - 构造方法之间调用 `this(...)`
- `this`
  - `this.xxx`
    - 特殊引用，引用当前对象的地址
  - `this(...)`
- 方法重载 `Overload`
- 继承
  - 目的：代码重用、复用

- 单继承
- 子类对象
  - 父类对象，和子类对象绑定，整体作为一个对象
  - 调用成员时，先找子类，再找父类
- 方法重写 Override
  - 继承的方法，在子类中重新定义，重新编写
  - super.xxxx()
    调用父类中同一个方法的代码

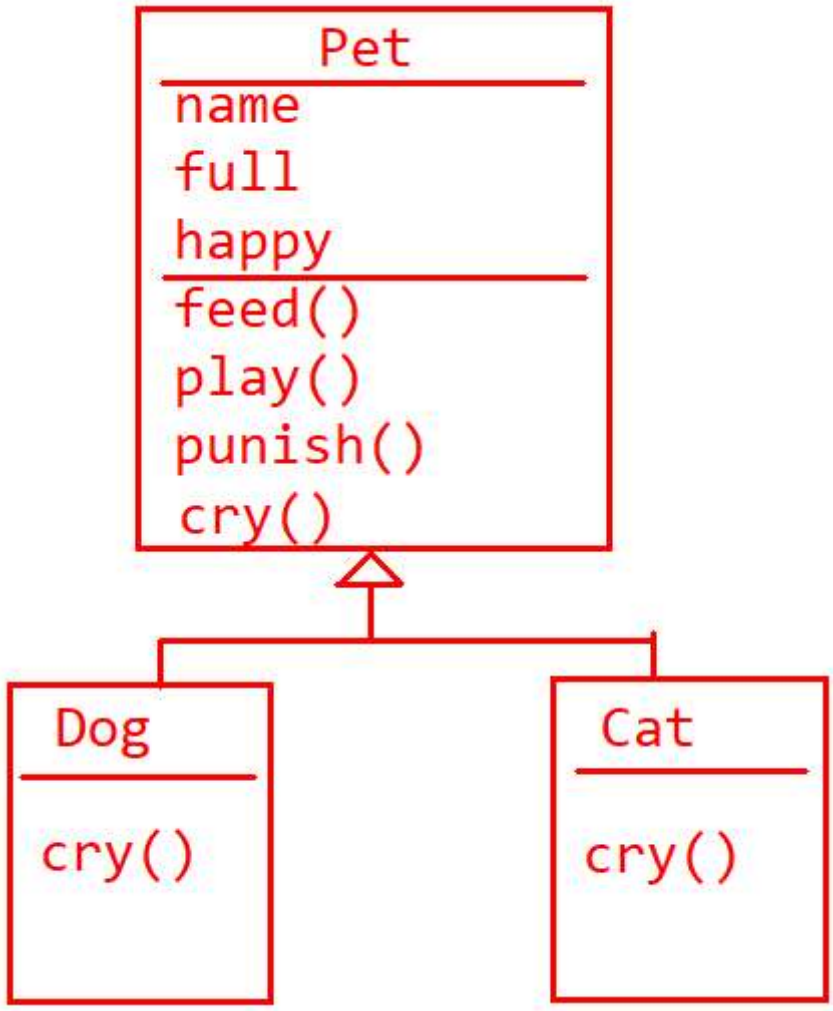
2 继承（续）

- 父类构造方法的执行
  - 新建子类对象时，会先新建父类对象
  - 也会先执行父类构造方法
  - 默认执行父类无参构造
    - super() 隐含调用
  - 手动调用调用父类的有参构造
    - super(参数)

电子宠物

添加 Cat 类

day0603_电子宠物
复制
day0701_电子宠物



Pet

```
package day0701;

public class Pet {

   String name;//null
   int full;//0
   int happy;//0

   public Pet(String name) {
      //有同名的局部变量
      //必须用this.name来访问成员变量
      this.name = name;
      full = 50;
      happy = 50;
   }
   public Pet(String name,int full,int happy) {
      this.name = name;
      this.full = full;
      this.happy = happy;
   }

   public void feed() {
```

```
        if(full == 100) {
            System.out.println(name+"已经吃不下了");
            return;
        }
        System.out.println("给"+name+"喂食");
        full += 10;
        System.out.println("饱食度: "+full);
    }

    public void play() {
        if(full == 0) {
            System.out.println(name+"饿得玩不动了");
            return;
        }
        System.out.println("陪"+name+"玩耍");
        happy += 10;
        full -= 10;
        System.out.println(
         "饱食度: "+full+", 快乐度: "+happy);
    }
    public void punish() {
        System.out.println(
         "打"+name+"的pp, "+name+"哭叫: "+cry());
        happy -= 10;
        System.out.println("快乐度: "+happy);
    }

    public String cry() {
        //无意义代码
        //需要在子类中重写
        return "此处有哭叫声";
    }

}
```

Dog

```
package day0701;
/*
 * 封装
 * 电子宠物的属性数据, 运算方法,
 * 封装成Dog类
 */
public class Dog extends Pet {
    public Dog(String name) {
        super(name);
    }

    @Override
```

```
    public String cry() {
        return "汪~";
    }
}
```

Cat

```
package day0701;

public class Cat extends Pet{
    public Cat(String name) {
        super(name);
    }

    @Override
    public String cry() {
        return "喵~";
    }
}
```

Test2

```
package day0701;

import java.util.Random;
import java.util.Scanner;

public class Test2 {
    public static void main(String[] args) {
        System.out.println("1.猫");
        System.out.println("2.狗");
        System.out.print("选择: ");
        int c = new Scanner(System.in).nextInt();
        System.out.print("给宠物起个名字: ");
        String name = new Scanner(System.in).nextLine();
        Cat cat = null;
        Dog dog = null;
        if(c==1) {
            cat = new Cat(name);
            play(cat);
        } else {
            dog = new Dog(name);
            play(dog);
        }
    }

    private static void play(Dog dog) {
        System.out.println("按回车继续");
        while(true) {
            new Scanner(System.in).nextLine();
```

```
            int r = new Random().nextInt(3);
            switch(r) {
            case 0: dog.feed(); break;
            case 1: dog.play(); break;
            case 2: dog.punish(); break;
            }
        }
    }
    private static void play(Cat cat) {
        System.out.println("按回车继续");
        while(true) {
            new Scanner(System.in).nextLine();
            int r = new Random().nextInt(3);
            switch(r) {
            case 0: cat.feed(); break;
            case 1: cat.play(); break;
            case 2: cat.punish(); break;
            }
        }
    }
}
```

3 多态

- 作用：一致的类型
  多种子类对象，都可以被当做一致的父类型来处理

- 类型转换
  - 向上转型
    子类型对象，转成父类型

    - 向上转型后，只能调用父类定义的通用成员
    - 子类特有的成员不能调用

  - 向下转型
    已经转为父类型的子类对象，再转回成子类型

- `Dog a = new Dog();`
  `Pet b = a;`

  引用变量类型的转换，对象本身不变

- `instanceof`
  运行期类型识别
  识别一个对象是否是指定的类型
  对真实类型，及其父类型判断，都返回 `true`

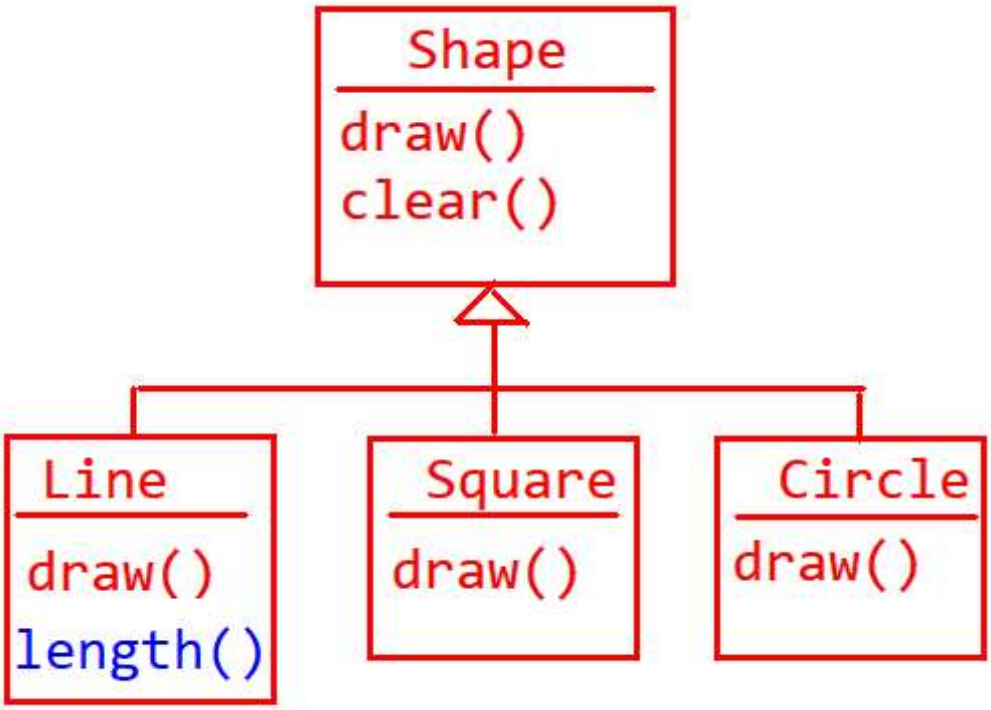
```
Shape s = new Line();

s instanceof Line     true
s instanceof Shape    true
```

图形形状

```
Shape
  |- Line
  |- Square
  |- Circle

项目：day0702_图形形状
类：day0702.Test1
            Shape
            Line
            Square
            Circle
```



Shape

```
package day0702;

public class Shape {
   public void draw() {
      //无意义代码
      //子类中需要重写这个方法
      System.out.println("图形形状");
   }
   public void clear() {
      System.out.println("\n\n\n\n\n\n\n\n\n\n");
```

```
    }
}
```

Line

```
package day0702;

public class Line extends Shape {
    @Override
    public void draw() {
        System.out.println("—");
    }
}
```

Square

```
package day0702;

public class Square extends Shape {
    @Override
    public void draw() {
        System.out.println("口");
    }
}
```

Circle

```
package day0702;

public class Circle extends Shape {
    @Override
    public void draw() {
        System.out.println("O");
    }
}
```

Test1

```
package day0702;

import java.util.Random;
import java.util.Scanner;

public class Test1 {
    public static void main(String[] args) {
        System.out.println("按回车执行");
        while(true) {
            new Scanner(System.in).nextLine();
```

```
            int r = new Random().nextInt(4);
            switch(r) {
            case 0: f(new Shape()); break;
            case 1: f(new Line()); break;
            case 2: f(new Square()); break;
            case 3: f(new Circle()); break;
            }
        }
    }

    private static void f(Shape shape) {
        shape.draw();
        new Scanner(System.in).nextLine();
        shape.clear();
    }
}
```

4 什么是面向对象

- 封装、继承、多态
- 封装
  - 类
  - 对象
  - 引用
  - 构造方法
  - this
  - 重载
- 继承
  - 作用：代码重用、复用
  - 子类对象
    - 父类对象和子类对象绑定
    - 调用成员时，先找子类再找父类
  - 重写
  - 父类构造方法
    - 默认 super()
    - 手动调用 super(参数)
- 多态
  - 向上、向下转型
  - instanceof

5 抽象类

- 半成品类，没有写完的类
- 抽象方法，只有方法定义，没有方法代码块
- 包含抽象方法的类，必须是抽象类
- 抽象类，不能创建对象
- 具体子类中，必须实现抽象方法
- 抽象方法的作用：
  - 作为通用方法，在父类中定义
  - 要求子类，必须完成这个方法

猜游戏

```
已经产生了5个不重复的大写字母
请猜这5个字母
    HCQRK
猜：HQKGK
2A2B
猜：RRRRR
1A4B
猜：HCQRK
5A0B

1.猜数字
2.猜字母
选择：

项目：day0703_猜游戏
类：day0703.Test1
            GuessGame
               |- GuessNumberGame
               |- GuessLetterGame
```

GuessGame

```
package day0703;

import java.util.Scanner;

public abstract class GuessGame {
   public void start() {
      //两种游戏的通用流程
      //产生随机值
      String r = suiJi();
      //显示提示
      tiShi();
      while(true) {
         System.out.print("猜：");
         String c = new Scanner(System.in).nextLine();
         //比较c和r，并得到比较的结果
         String result = biJiao(c, r);
```

```
            System.out.println(result);
            //result是否是猜对的结果
            if(caiDui(result)) {
               break;
            }
         }

      }

      public abstract String suiJi();
      public abstract void tiShi();
      public abstract String biJiao(String c, String r);
      public abstract boolean caiDui(String result);
}
```

GuessNumberGame

```
package day0703;

import java.util.Random;

public class GuessNumberGame extends GuessGame {
   @Override
   public String suiJi() {
      int r = 1 + new Random().nextInt(1000);
      //把整数，变成字符串类型
      return String.valueOf(r);
   }
   @Override
   public void tiShi() {
      System.out.println(
       "已经随机产生了一个[1,1001)范围的整数");
      System.out.println("请猜这个数是几");
   }
   @Override
   public String biJiao(String c, String r) {
      //字符串解析成int
      int cc = Integer.parseInt(c);
      int rr = Integer.parseInt(r);
      if(cc>rr) {
         return "大";
      } else if(cc<rr) {
         return "小";
      } else {
         return "对";
      }
   }
   @Override
```

```
    public boolean caiDui(String result) {
        return "对".equals(result);
    }
}
```

GuessLetterGame

```
package day0703;

public class GuessLetterGame extends GuessGame {
    @Override
    public String suiJi() {
        // TODO: 稍后再完成
        return "ABCDE";
    }
    @Override
    public void tiShi() {
        System.out.println("已经产生了5个不重复的大写字母");
        System.out.println("请猜这5个字母");
    }
    @Override
    public String biJiao(String c, String r) {
        // TODO: 稍后再完成
        return "2A2B";
    }
    @Override
    public boolean caiDui(String result) {
        return "5A0B".equals(result);
    }
}
```

Test1

```
package day0703;

import java.util.Scanner;

public class Test1 {
    public static void main(String[] args) {
        System.out.println("1. 猜数字");
        System.out.println("2. 猜字母");
        System.out.println("选择: ");
        int c = new Scanner(System.in).nextInt();
        GuessGame game;
```

```
        if(c == 1) {
            game = new GuessNumberGame();
        } else {
            game = new GuessLetterGame();
        }

        game.start();
    }
}
```

6 窗口

- javax.swing.JPanel
  封装面板的基础代码
- 自定义面板类，从JPanel继承基础代码，再添加要显示的内容

面板

项目：day0704_面板
类：day0704.Test1
MyPanel

MyPanel

```
package day0704;

import java.awt.Graphics;

import javax.swing.JPanel;

public class MyPanel extends JPanel {
    /*
     * 固定的绘制方法
     * 面板被绘制时，或被重绘时，
     * 系统会自动调用这个方法
     *
     * 参数g：是一张画布
     */
```

```
    @Override
    public void paint(Graphics g) {
        g.fillOval(100, 100, 80, 40);
        g.drawLine(50, 50, 200, 150);
        g.drawRect(20, 80, 180, 120);
    }
}
```

Test1

```
package day0704;

import javax.swing.JFrame;

public class Test1 {
    public static void main(String[] args) {
        JFrame f = new JFrame();
        f.setSize(400, 400);
        f.setTitle("自定义面板");
        f.setDefaultCloseOperation(
          JFrame.EXIT_ON_CLOSE);

        //面板对象
        MyPanel panel = new MyPanel();
        //在窗口f中，添加面板panel
        f.add(panel);

        f.setVisible(true);
    }
}
```

7 作业

- 重写
  - day0701_电子宠物
    - Pet
    - Dog
    - Cat
    - Bird
    - Fish
    - Test2
  - day0703_猜游戏
    - GuessGame
    - GuessNumberGame
    - GuessLetterGame

- ◆ Test1

- 复习面向对象概念