目录

Day09.Java

1 回顾

- 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
  - private
- 继承
  - 作用：代码重用、复用
  - 子类对象
    - 由父类对象和子类对象绑定，整体作为一个对象
    - 调用成员，先找子类，再找父类
  - 重写
    - 继承的方法，在子类中，重新定义，重新编写（改写）
  - 父类构造方法
    - 默认 super()

        - ◆ 手动调用 super(参数)
    - super
        - ◆ super.xxxx()
          重写时，调用父类同一个方法的代码
        - ◆ super(...)

- 多态
    - 作用：一致的类型
    - 向上转型
    - 向下转型
    - instanceof

- 抽象类
- final（3种用法）
    - 常量
    - 方法，不能重写
    - 类，不能继承
- static
    - 静态，属于类
    - 不是面向对象语法
    - 使用场景
        - ◆ 共享的数据
        - ◆ 工具方法
          Math.random()
          String.valueOf()
          Integer.parseInt()
    - static {
      }
- 常量
    - static final
    - 全大写，单词用下划线连接
- 对象创建过程（10步）

2 接口

- 作用：结构设计工具，用来解耦合
- 极端的抽象类，所有的方法都是抽象方法
- 用 interface 代替 class 关键字
- 用 implements 代替 extends 关键字
- 接口中只能定义：
    - 公开的常量
    - 公开的抽象方法
    - 公开的内部类、内部接口
- 一个类可以同时实现多个接口

```
class A implements X,Y,Z {
}
```

```
class A extends B implements X,Y,Z {
}
```

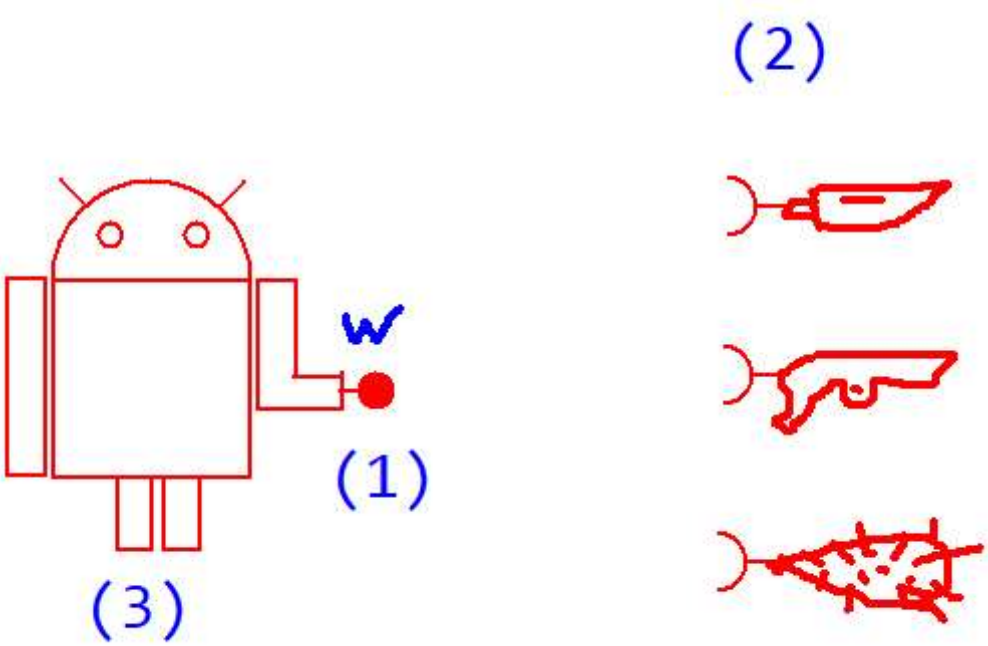
- 接口之间继承

```
interface A extends X,Y,Z {
}
```

变形金刚

项目：day0901_变形金刚
类：day0901.Test1

```
            Weapon
               |- Sword
               |- AK47
               |- Lyb
            Transformer
```



Weapon

```
package day0901;

public interface Weapon {
   /* public static final */
   int TYPE_COLD      =0;
   int TYPE_HOT    =1;
   int TYPE_NUCLEAR   =2;

   /* public abstract */
   void kill();
```

```
    String getName();
    int getType();
}
```

Sword

```
package day0901;

public class Sword implements Weapon {
    @Override
    public void kill() {
        System.out.println("耍剑");
    }
    @Override
    public String getName() {
        return "倚天剑";
    }
    @Override
    public int getType() {
        return Weapon.TYPE_COLD;
    }
}
```

AK47

```
package day0901;

public class AK47 implements Weapon {
    @Override
    public void kill() {
        System.out.println("耍枪");
    }
    @Override
    public String getName() {
        return "AK47";
    }
    @Override
    public int getType() {
        return Weapon.TYPE_HOT;
    }
}
```

Lyb

```
package day0901;

public class Lyb implements Weapon {
    @Override
    public void kill() {
        System.out.println("耍棒");
    }
```

```
    @Override
    public String getName() {
        return "狼牙棒";
    }
    @Override
    public int getType() {
        return Weapon.TYPE_NUCLEAR;
    }
}
```

Transformer

```
package day0901;

public class Transformer {
    //变形金刚使用的武器接口
    //定义武器接口类型的变量
    private Weapon w; // null
    public void setWeapon(Weapon w) {
        this.w = w;
    }

    public void attack() {
        //武器接口上没有接入武器
        if(w == null) {
            System.out.println("用牙咬");
            return;
        }

        //使用冷兵器倚天剑进攻
        String type = null;
        switch(w.getType()) {
        case Weapon.TYPE_COLD: type="冷兵器"; break;
        case Weapon.TYPE_HOT: type="热兵器"; break;
        case Weapon.TYPE_NUCLEAR: type="核武器"; break;
        }

        System.out.println(
         "使用"+type+w.getName()+"进攻");

        w.kill();
    }
}
```

Test1

```
package day0901;

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

public class Test1 {
   public static void main(String[] args) {
      Transformer t = new Transformer();
      Sword s = new Sword();
      AK47 a = new AK47();
      Lyb l = new Lyb();
      System.out.println("按回车继续");
      while(true) {
         new Scanner(System.in).nextLine();
         int r = new Random().nextInt(4);
         switch(r) {
         case 0: t.setWeapon(s); break;
         case 1: t.setWeapon(a); break;
         case 2: t.setWeapon(l); break;
         case 3: t.setWeapon(null); break;
         }
         t.attack();
      }
   }
}
```

3 内部类

- 嵌套定义在类中、方法中、局部代码块中
- 非静态内部类
- 静态内部类
- 局部内部类
- 匿名内部类

3.1 非静态内部类

```
class A {
   class Inner {
   }
}

A a = new A();
```

```
Inner i = a.new Inner();
```

- 一般用来封装一个复杂对象中的局部数据，或局部的运算逻辑

3.2 静态内部类

```
class A {
   static class Inner {
   }
}

Inner i = new Inner();
```

- 与普通的类相同
- 静态内部类是一个设计的可选项
- 考虑两个类的关系紧密，可以选择嵌套定义

3.3 局部内部类

```
class A {
   Weapon f() {
      class Inner implements Weapon {
      }

      Inner i = new Inner();
      return i;
   }
}

Weapon w = a.f();
```

3.4 匿名内部类

```
Weapon w = new Weapon() {...};
```

- 大括号是匿名类
- new 新建匿名类的对象
- 小括号 super()、super(参数)
- 变量类型必须用父类型

内部类

项目：day0902_内部类
类：day0902.Test1

Test1

```
package day0902;

import day0902.A.Inner1;
import day0902.A.Inner2;

public class Test1 {
   public static void main(String[] args) {
      Inner1 i1 = new Inner1();
      System.out.println(i1);

      A a = new A();
      Inner2 i2 = a.new Inner2();
      System.out.println(i2);
   }
}

class A {
   static class Inner1 {
   }

   class Inner2 {
   }
}
```

Test2

```
package day0902;

import java.util.Scanner;

public class Test2 {
   public static void main(String[] args) {
      Weapon w = f1();
      w.kill();

      System.out.println("------------");

      System.out.print("使用的武器名称: ");
      String n = new Scanner(System.in).nextLine();

      Weapon w2 = new Weapon(){
         @Override
         public void kill() {
            //局部内部类中，使用外面的局部变量
            //必须加final
```

```
            //jdk1.8后，缺省存在
            System.out.println("使用"+n+"进攻");
        }
    };

    f2(w2);

}


static void f2(Weapon w) {
    w.kill();
}



private static Weapon f1() {
    class AK47 implements Weapon {
        @Override
        public void kill() {
            System.out.println("使用AK47进攻");
        }
    }

    AK47 a = new AK47();
    return a;
}
}
```

4 飞机大战 - 英雄机

- 英雄机随鼠标移动

day0806_飞机大战
复制
day0903_飞机大战

Hero
属性:
    imgs
    img
    x
    y
    tx 目标位置

```
    ty

方法:
    paint(g)
    setTarget(x,y)
    step() 移动一帧
```

Hero

```
package day0804;

import java.awt.Graphics;
import java.awt.image.BufferedImage;

public class Hero {
    BufferedImage[] imgs = Main.hero;
    BufferedImage img = imgs[1];

    int x = 152;
    int y = 480;
    //目标位置
    int tx;
    int ty;

    public void paint(Graphics  g) {
        g.drawImage(img, x, y, null);
    }
    /*
     * 当鼠标移动时，设置英雄机移动的目标位置
     * 参数x,y是鼠标的位置坐标
     */
    public void setTarget(int x, int y) {
        tx = x-48;
        ty = y-62;
    }

    public void step() {
        //目标位置和英雄机当前位置
        //是否是同一个位置
        if(x==tx && y==ty) {
            img = imgs[1];//静止图片
        } else {
            x = tx;
            y = ty;
            img = imgs[0];//加速图片
        }
    }
}
```

GamePanel

```
package day0804;

import java.awt.Dimension;
import java.awt.Graphics;
import java.awt.event.MouseAdapter;
import java.awt.event.MouseEvent;

import javax.swing.JPanel;

public class GamePanel extends JPanel {
    Background bg = new Background();
    Enemy[] ememys = {
            new Airplane(),
            new BigPlane(),
            new Bee()
    };
    Hero hero = new Hero();

    public GamePanel() {
        //设置面板的期望大小
        setPreferredSize(new Dimension(400, 654));
    }

    /*
     * 固定的绘图方法
     * 由系统自动调用
     */
    @Override
    public void paint(Graphics g) {
        bg.paint(g);
        for (int i = 0; i < ememys.length; i++) {
            Enemy e = ememys[i];
            e.paint(g);
        }
        hero.paint(g);
    }

    //动起来方法
    public void action() {
        //设置鼠标监听器
        addMouseMotionListener(new MouseAdapter() {
            @Override
            public void mouseMoved(MouseEvent e) {
                hero.setTarget(e.getX(), e.getY());
            }
        });


        //画面一帧一帧的循环播放
        while(true) {
            bg.step();//背景移动
            for (int i = 0; i < ememys.length; i++) {
                Enemy e = ememys[i];
```

```
                e.step();
            }
            hero.step();

            //通知底层系统，重绘画面
            //系统受到通知后，会自动调用 paint() 方法
            repaint();
            //暂停 1/60 秒 (60 fps)
            try {
                Thread.sleep(1000/60);
            } catch (InterruptedException e) {
            }
        }
    }
}
```

5 基础API

- API
  Application Progrmming Interface
  应用编程接口
- 一切可以调用的东西

- java.lang 包
  - lang = language
  - lang包所有的类，自动导入
    String
    System

- Object
- String
- StringBuilder/StringBuffer
- 正则表达式
- 基本类型包装类
- BigDecimal/BigInteger
- Date
- SimpleDateFormat

6 Object

- 顶层父类

- 一个类，如果不继承其他类，默认继承Object

```
class A /*extends Object*/ {

}
```

- 方法
  - toString()
    获得一个对象的字符串表示
    Object 中默认实现：
    "day0904.Point@81fa9c2"

    如果需要自定义字符串格式，可以在子类中重写这个方法

  - equals(Object obj)
    当前对象，与参数对象obj比较是否相等
    Object 中默认实现：比较内存地址

    ```
    this == obj
    ```

    如果要比较对象中的属性是否相等，可以在子类中，重写 equals() 方法

Object

项目：day0904_Object
类：day0904.Test1
Point

Point

```
package day0904;

public class Point /*extends Object*/ {
   private int x;
   private int y;

   public Point() {
      super();
   }
   public Point(int x, int y) {
      super();
      this.x = x;
      this.y = y;
   }
   public int getX() {
      return x;
   }
   public void setX(int x) {
      this.x = x;
   }
   public int getY() {
      return y;
```

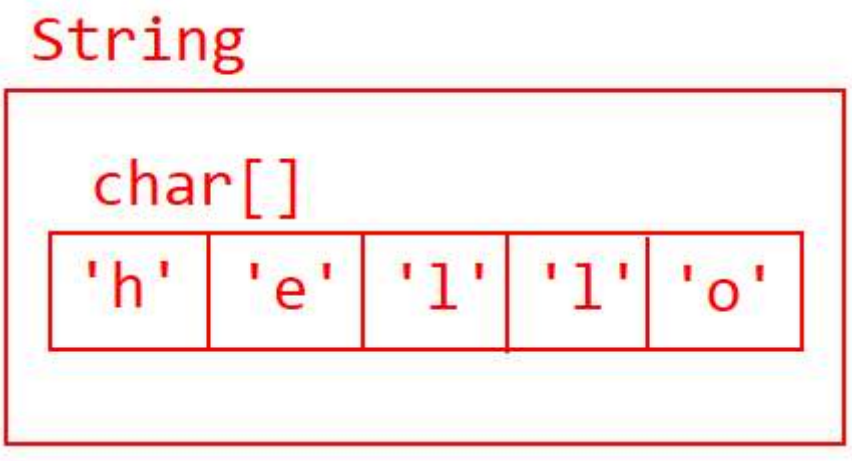
```
    }
    public void setY(int y) {
        this.y = y;
    }

    @Override
    public String toString() {
        return "("+x+", "+y+")";
    }

    @Override
    public boolean equals(Object obj) {
        //参数的三种特殊情况
        if(obj == null) return false;
        if(obj == this) return true;
        if(! (obj instanceof Point)) return false;

        //把obj向下转回成 Point 类型
        Point p = (Point) obj;
        return x == p.x &&
               y == p.y;
    }
}
```

Test1

```
package day0904;

public class Test1 {
    public static void main(String[] args) {
        Point a = new Point(3, 4);
        Point b = new Point(3, 4);

        System.out.println(a.toString());
        System.out.println(b);

        System.out.println(a == b);
        System.out.println(a.equals(b));
    }
}
```

7 String

- 封装 char[] 数组的对象



- 创建对象
  - char[] a = {'h', 'e', 'l', 'l', 'o'};
    String s = new String(a);
  - 第一种语法的语法简化:
    String s = "hello";
- 字符串常量池
  - 第一次使用一个字符串的字面值
    会在“常量池”新建对象
  - 再次使用相同字面值时，直接访问常量池中存在的对象，而不新建
- 字符串不可变
  - String s1 = "aaa";
    String s2 = "bbb";
    String s3 = "ccc";
    String s4 = s1 + s2 + s3;
  - 字符串加号连接，效率低
    - 会新建字符串对象

字符串

项目：day0905_字符串
类：day0905.Test1

Test1

```
package day0905;

public class Test1 {
   public static void main(String[] args) {
      char[] a = {'h','e','l','l','o'};
      String s1 = new String(a);//新分配内存

      String s2 = "hello";//常量池新分配内存
```

```
        String s3 = "hello";//访问常量池存在的对象

        System.out.println(s1 == s2);
        System.out.println(s2 == s3);
        System.out.println(s1.equals(s2));
    }
}
```

Test2

```
package day0905;

public class Test2 {
    public static void main(String[] args) {
        String s0 = "abcdefghijklmnopqrstuvwxyz";
        String s = "";
        //获得系统当前时间毫秒值
        //毫秒值：从1970-1-1 0点开始的毫秒值
        long t = System.currentTimeMillis();
        for(int i=0;i<100000;i++) {
            s += s0;
        }
        t = System.currentTimeMillis()-t;
        System.out.println(t);
    }
}
```

8 作业

- 重写
  - day0901_变形金刚
    - Weapon
    - Sword
    - AK47
    - Lyb
    - Transformer
    - Test1
  - 飞机大战-英雄机
    - day0806_飞机大战
      复制一个新项目
    - Hero
    - GamePanel
      - 新建Hero对象
      - 绘制英雄
      - 英雄移动一帧
      - 鼠标监听器

- Student类
  - id,name,gender,age
  - 重写 toString()和equals()
  - id,age，int类型，用==
  - name,gender，String类型，用equals()