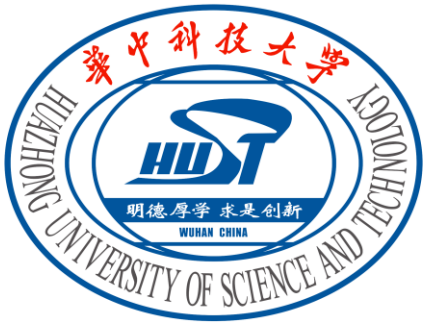
基于Java的面向对象程序设计

陈维亚

weiya_chen@hust.edu.cn

华中科技大学软件学院

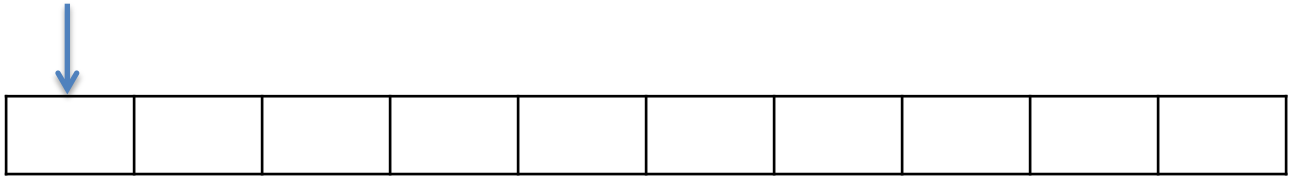
第12讲：综合练习



目录

1. 数组专题

❑ 一维数组



```
// 1st way
int[] nums1 = new int[10];

// 2nd way
double[] nums2 = {0.0, 0.0, 0.0};

for(int i=0; i<nums1.length; i++){
  System.out.println(nums[i]);
}
```

1. 数组专题

❑ 一维数组

```
public class Cat {
  private String name;
  private int age;
}
```

```
Cat[] cats = new Cat[10];

for(int i=0; i<cats.length; i++){
  System.out.println(cats[i].getName());
}
```

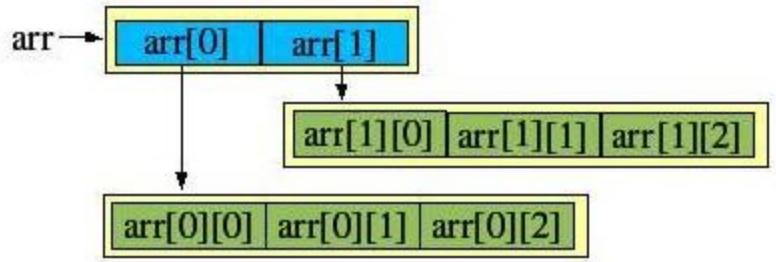
NullPointerException

```
Cat[] cats = new Cat[10];

for(int i=0; i<cats.length; i++){
  cats[i] = new Cat("Kitty", 100);
  System.out.println(cats[i].getName());
}
```

1. 数组专题

❑ 二维数组



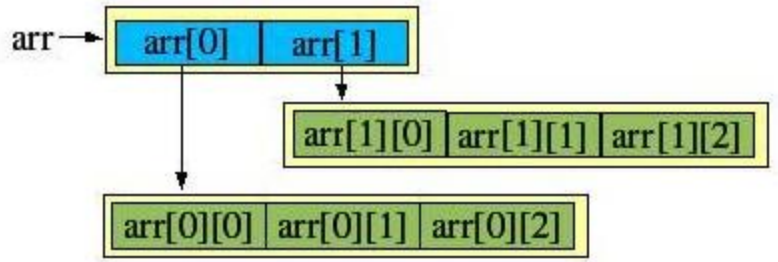

```
int[][] arr = new int[2][3];

int[][] mat = {{0,1,2}, {1,2,3}};

for(int i=0; i<mat.length; i++){
    for(int j=0; j<mat[i].length; j++){
        System.out.println(mat[i][j]);
    }
}
```

1. 数组专题

❑ 二维数组




```
Cat[][] mat = {{new Cat(),new Cat()},
               {new Cat(),new Cat()}};

for(int i=0; i<mat.length; i++){
    for(int j=0; j<mat[i].length; j++){
        System.out.println(mat[i][j].getName());
    }
}
```

2. 方法专题

❑ 方法的声明

```
public (static) 返回值类型 方法名[形式参数列表];
```

命名遵循驼峰规则：getName()

❑ 方法的定义

```
public (static) 返回值类型 方法名[形式参数列表]{
    方法体
}
```

返回值不为空时，必须用return

2. 方法专题

❑ 构造方法

```
public 类名[形式参数列表]；
```

用于对象的初始化（创造一个类的实例）；

成员变量的赋值一般都在构造函数中进行；

注意父类的构造函数调用规则；

```
public class Cat {
    private Tail tail;

    public Cat(){
        tail = new Tail();
    }
}
```

2. 方法专题

❑ 方法的重载 overload

同名不同参

可在同一个类中出现

❑ 方法的重写 override

同名又同参，又同返回值（一模一样）

子类与父类中出现

3. 类与对象专题

❑ 类的定义

一个文件里只能有一个public类

此文件名必须和public类名相同

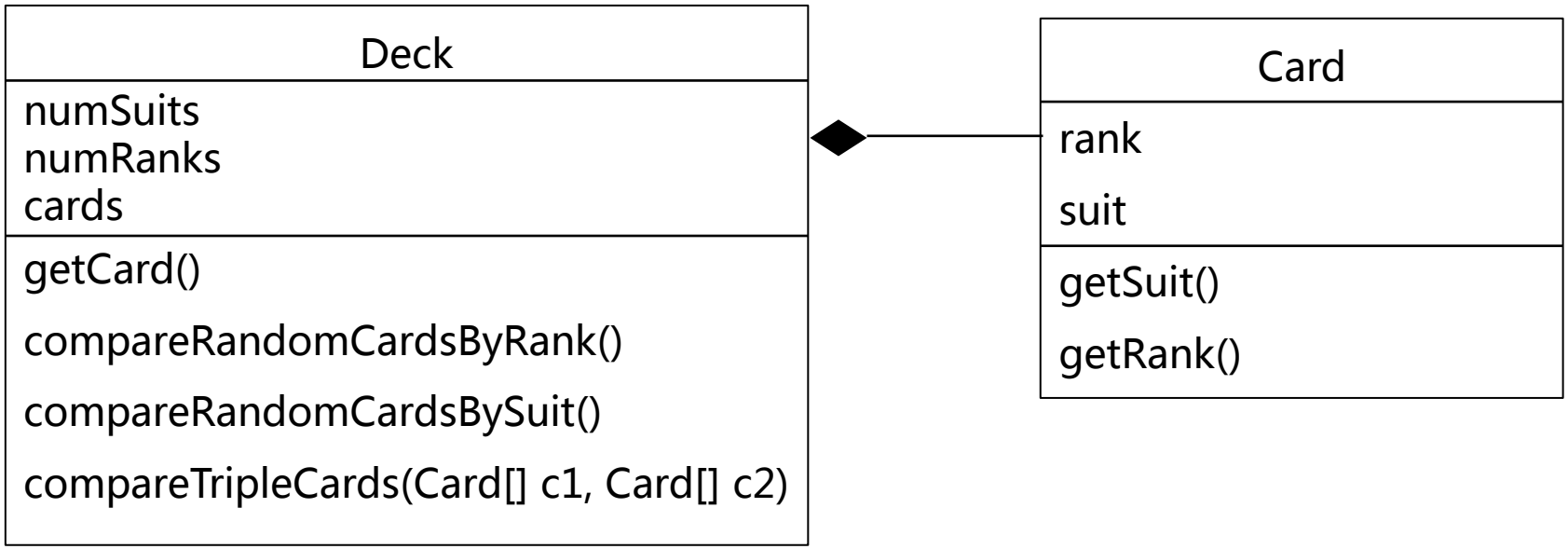
类里只能定义属性（成员变量）和方法（成员方法和静态方法）

❑ 对象的初始化

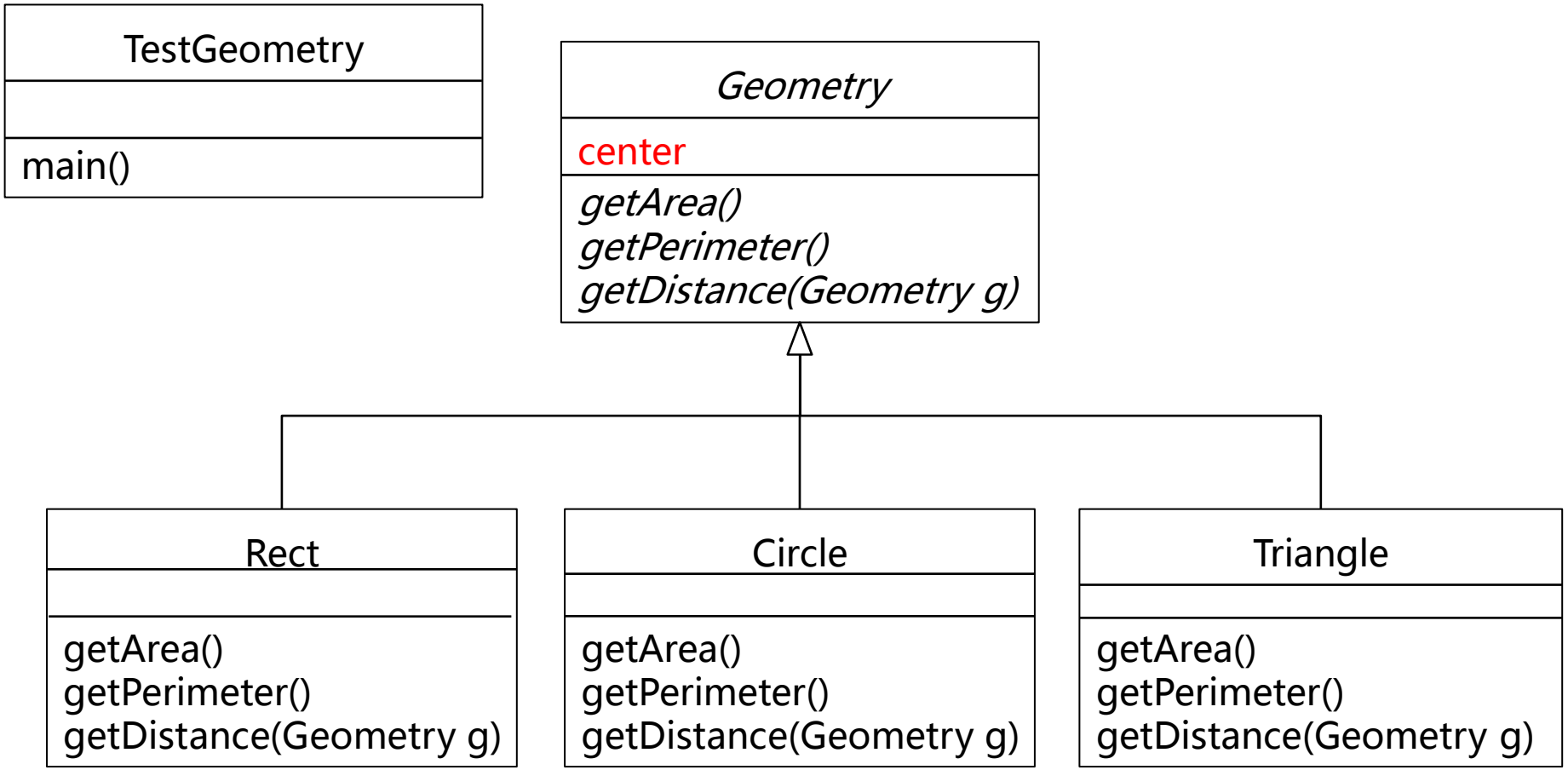
```
Cat cat = new Cat();

System.out.println(cat.setName("Tom"));
```

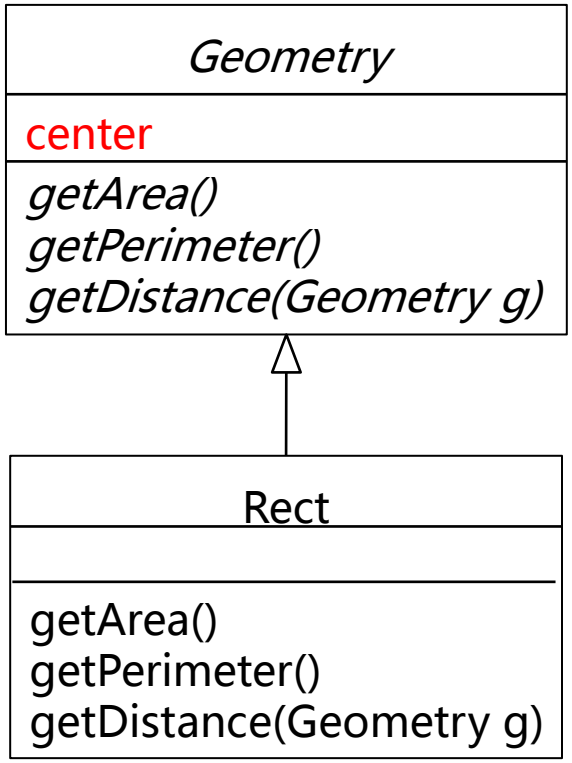
4. 上机总结


Deck
numSuits
numRanks
cards
getCard()
compareRandomCardsByRank()
compareRandomCardsBySuit()
compareTripleCards(Card[] c1, Card[] c2)
Card
rank
suit
getSuit()
getRank()

4. 上机总结


TestGeometry
main()
Geometry
center
getArea()
getPerimeter()
getDistance(Geometry g)
Rect
getArea()
getPerimeter()
getDistance(Geometry g)
Circle
getArea()
getPerimeter()
getDistance(Geometry g)
Triangle
getArea()
getPerimeter()
getDistance(Geometry g)

4. 上机总结



Rect：

```
public double getDistance(Geometry g){
    double dx = this.center.getX() -
g.getCenter().getX();
    double dy = this.center.getY() -
g.getCenter().getY();
    return Math.sqrt(dx*dx + dy*dy);
}
```

```
Geometry g1 = new Rect(100, 80, 10, 10);
Geometry g2 = new Circle(50, 60, 200);

System.out.println(g1.getDistance(g2));
```

若在Geometry类中增加比较周长的方法，如何在子类中实现？

```
public abstract int hasLongerPerimeter(Geometry g);
```

下节预告

类的关联