异常捕获

- 自动捕获 try-cath
- throw 主动抛出异常
- throws 修饰可能抛出异常的方法

自定义异常

除了使用 Java 提供的异常外，也可以根据需求来自定义异常。

```
package com.southwind.exception;

public class MyNumberException extends RuntimeException {
  public MyNumberException(String error) {
    super(error);
  }
}
```

```
package com.southwind.exception;

public class Test {
  public static void main(String[] args){
    Test test = new Test();
    System.out.println(test.add("a"));
  }

  public int add(Object object){
    if(object instanceof Integer) {
      int num = (int)object;
      return ++num;
    }else {
      String error = "传入的参数不是整数类型";
      MyNumberException myNumberException = new MyNumberException(error);
      throw myNumberException;
    }
  }
}
```

综合练习

封装、继承、多态、抽象、接口、异常完成一个汽车查询系统。

需求描述：共有 3 种类型的汽车：小轿车、大巴车、卡车，其中小轿车的座位数是 4 座，大巴车座位数是 53 座，卡车座位数是 2 座，要求使用封装、继承、抽象来完成车辆的定义。

可以对车辆信息进行修改，卡车可以运货但是载重量不能超过 12 吨，使用自定义异常来处理错误，小轿车和大巴车没有此功能，要求使用接口来实现。

Car

```java
package com.southwind.test;

public abstract class Car {
  private String name;
  private String color;
  public String getName() {
    return name;
  }
  public void setName(String name) {
    this.name = name;
  }
  public String getColor() {
    return color;
  }
  public void setColor(String color) {
    this.color = color;
  }
  public Car(String name, String color) {
    super();
    this.name = name;
    this.color = color;
  }
  public abstract String seatNum();
}
```

Sedan

```java
package com.southwind.test;

public class Sedan extends Car {

  public Sedan(String name, String color) {
    super(name, color);
  }

  @Override
  public String seatNum() {
    // TODO Auto-generated method stub
    return "4座";
  }

}
```

Bus

```
package com.southwind.test;

public class Bus extends Car {

  public Bus(String name, String color) {
    super(name, color);
    // TODO Auto-generated constructor stub
  }

  @Override
  public String seatNum() {
    // TODO Auto-generated method stub
    return "53座";
  }

}
```

Truck

```
package com.southwind.test;

public class Truck extends Car implements Container {

  private int weight;

  public Truck(String name, String color,int weight) {
    super(name, color);
    this.weight = weight;
    // TODO Auto-generated constructor stub
  }

  @Override
  public String seatNum() {
    // TODO Auto-generated method stub
    return "2座";
  }

  @Override
  public int getweight() {
    // TODO Auto-generated method stub
    return this.weight;
  }

}
```

Container

```
package com.southwind.test;

public interface Container {
  public int getweight();
}
```

CarException

```
package com.southwind.test;

public class CarException extends Exception {
  public CarException(String error) {
    super(error);
  }
}
```

Test

```
package com.southwind.test;

import java.util.Scanner;

public class Test {
  private static Scanner scanner;
  private static Sedan sedan;
  private static Bus bus;
  private static Truck truck;
  private static Car[] cars;
  static {
    scanner = new Scanner(System.in);
    sedan = new Sedan("小轿车","黑色");
    bus = new Bus("大巴车","绿色");
    truck = new Truck("卡车","蓝色",2);
    cars = new Car[3];
    cars[0] = sedan;
    cars[1] = bus;
    cars[2] = truck;
  }

  public void showCars() {
    System.out.println("欢迎使用本汽车管理系统");
    System.out.println("车辆名称\t\t车辆颜色\t\t座位数\t\t载重量");
    for(Car car:cars) {
      if(car instanceof Truck) {
        Truck truck = (Truck)car;

System.out.println(car.getName()+"\t\t"+car.getColor()+"\t\t"+car.seatNum()+"\
t\t"+truck.getweight());
```

```
        }else {

System.out.println(car.getName()+"\t\t"+car.getColor()+"\t\t"+car.seatNum()+"\
t\t不能拉货");
        }
      }
      System.out.println("1.小轿车\t2.大巴车\t3.卡车");
      System.out.print("请选择要修改的车辆: ");
      int num = scanner.nextInt();
      switch(num) {
        case 1:
          update("sedan");
          break;
        case 2:
          update("bus");
          break;
        case 3:
          update("truck");
          break;
        default:
          System.out.println("车辆不存在!");
          break;
      }
    }

    public void update(String type) {
      String name = null;
      String color = null;
      if(type.equals("sedan")) {
        System.out.print("输入车辆名称");
        name = scanner.next();
        System.out.print("输入车辆颜色");
        color = scanner.next();
        Sedan sedan = new Sedan(name,color);
        cars[0] = sedan;
      }
      if(type.equals("bus")) {
        System.out.print("输入车辆名称");
        name = scanner.next();
        System.out.print("输入车辆颜色");
        color = scanner.next();
        Bus bus = new Bus(name,color);
        cars[1] = bus;
      }
      if(type.equals("truck")) {
        System.out.print("输入车辆名称");
        name = scanner.next();
        System.out.print("输入车辆颜色");
        color = scanner.next();
        System.out.print("输入载重量");
        int weight = scanner.nextInt();
        if(weight > 12) {
          CarException carException = new CarException("卡车的载重量不能超过12吨");
          try {
            throw carException;
          } catch (CarException e) {
            // TODO Auto-generated catch block
            e.printStackTrace();
            return;
          }
        }
        Truck truck = new Truck(name,color,weight);
        cars[2] = truck;
      }
      showCars();
    }

    public static void main(String[] args) {
      Test test = new Test();
      test.showCars();
    }
  }
```

讲解了面向对象的高级部分，包括 Object 类、包装类、接口和异常。其中 Object 类是所有 Java 类的父类，定义了 Java 体系的基础资料，通过继承传递给 Java 的每一个类，通过方法重写和多态让整个 Java 体系具有很强的灵活性。

包装类是 Java 为基本数据类型提供封装的一组类，通过包装类我们可以将基本数据类型转为对象，这一点在面向对象编程中很重要。

接口是抽象类的扩展，是 Java 中实现多态的重要方式，可以降低程序的耦合性，让程序变得更加灵活多变。接口就相当于零件，我们可以自由地将这些零件进行组装、整合。

异常是 Java 中处理错误的一种机制，同样是基于面向对象的思想，将错误抽象成对象然后进行处理，这里需要关注的是对异常相关的几个关键字的使用，try、catch、finally、throw、throws。