第3章-类中的方法

选择结构

循环结构

打印九九乘法表

```
public class MultiTable{
    public static void main(String[] args){
        for(int i = 1;i <= 9; i++){
            for(int j = 1;j <= i;j++)
                System.out.print(" "+i+"*"+j+" = "+i*j);
            System.out.println();
        }
    }
}
```

增强for循环

```
public class PrintDay{
    public static void main(String[] args){
        String[] days = {"Monday", "Tuesday", "Wednesday", "Thursday",
"Friday", "Saturday", "Sunday"};
        for(String day : days){
            System.out.println(day);
        }
    }
}
```

break可与label一同使用

```
public class TestBreakLabel{
    public static void main(String[] args){
        outer:
        for(int i = 1;i <= 9;i++){
            for(int j = 1;j <= 9;j++){
                if(j > i)break;
                if(i == 6)break outer;
                System.out.print(" "+i+"*"+j+"="+i*j);
            }
            System.out.println();
        }
    }
}
```

continue可与label一同使用

```
public class TestContinueLabel{
    public static void main(String[] args){
        outer:
        for(int i = 1;i < 10;i++){
            inner:
```

```
            for(int j = 1;j < 10;j++){
                if(i < j){
                    System.out.println();
                    continue outer;
                }
                System.out.print(" "+i+"*"+j+"="+i*j);
            }
        }
    }
}
```

Java处理错误的方法

错误的概念及分类

异常分类

对于检查型异常，Java强迫程序必须进行处理。

处理方法有两种：

声明抛出异常：不在当前方法内处理异常，而是把异常抛出到调用方法中。

捕获异常：使用try{}catch{}块，捕获到所发生的异常，并进行相应的处理。

捕获异常语法格式，见PPT

说明：

try语句：其后跟随可能产生异常的代码块。

catch语句：其后跟随异常处理语句，通常用到两个方法，getMessage（）和printStackTrace（）

finally语句：不论在try代码段是否产生异常，finally后的程序代码段都会被执行。通常在这里释放内存以外的其他资源。

```
import java.io.*;
class Keyboard{
    static BufferedReader inputStream = new BufferedReader
    (new InputStreamReader(System.in));
    public static int getInteger() {
        try{
            return
(Integer.valueOf(inputStream.readLine().trim()).intValue());
        }
        catch (Exception e) { e.printStackTrace(); return 0; }
    }
    public static String getString() {
        try{
            return (inputStream.readLine());
        }
        catch (IOException e) { return "0";}
    }
}
public class ExceptionTester4{
    public static void main(String[] args){
        int num[] = new int[2];
        int result;
        boolean valid;
        for(int i = 0;i < 2;i++){
```

```
            valid = false;
            while(!valid){
                try{
                    System.out.println("Enter number" + (i+1));
                    num[i] =
Integer.valueOf(Keyboard.getString()).intValue();
                    valid = true;
                }catch(NumberFormatException e){
                    System.out.println("Invalid input.Please try again:");
                }
            }
        }
        try{
            result = num[0] / num[1];
            System.out.print(num[0] + "/" + num[1] + "=" + result);
        }catch(ArithmeticException e){
            System.out.println("Division by zero is not allowed");
        }
    }
}
```

声明自己的异常类：除使用系统预定义的异常类外，用户还可以声明自己的异常类。

自定义的所有异常类都必须是Exception的子类。

如果使用者可以从该异常中恢复，则定义为检查型异常；否则可以定义为非检查型异常（少）

方法签名：方法名和参数类型一起，构成方法签名**（注意不包括返回值类型）**

方法重载

联编（binding，绑定）就是把方法调用和方法体代码联结起来的过程。有两种形式的联编。

静态联编（又叫早期联编early binding）：编译时根据变量类型和方法名称，决定被调方法的签名。

动态联编（又叫晚期联编late binding）：编译时无法确定被调用的方法体，运行时根据对象类型决定。