

一、实验名称

数据类型与表达式部分

二、实验目的及要求

1. 第1章 **page 4** 基础训练2: 当同一文件中含两个类时，访问另一类的静态属性
目的: 了解如何访问另一个类的静态属性。
2. 第二章 **page10** 基础练习1: 变量的数据类型及赋值。
目的: 了解变量的赋值及转换问题。
3. 第二章 **page11** 基础练习2: 典型运算符的使用
目的: 熟悉++和%运算符的使用。

三、实验环境

调试工具:

IntelliJ IDEA 2024.3.2.2 (Ultimate Edition)

服务配置:

openjdk 23.0.2 2025-01-21

OpenJDK Runtime Environment (build 23.0.2+7-58)

OpenJDK 64-Bit Server VM (build 23.0.2+7-58, mixed mode, sharing)

四、实验设计

1. 第1章 **page 4** 基础训练2: 当同一文件中含两个类时，访问另一类的静态属性

```
public class My {  
    public static void main(String[] args) {  
        System.out.println(you.info);  
    }  
}  
  
class you {  
    static String info = "同学们好!";  
}
```

思考:

1. 编译产生了两个类文件, 分别叫 `My.class` 和 `you.class`

```
@29701 on E:/CODE/Java/HW/src/ex1 main base 3.12.7
# javac My.java
@29701 on E:/CODE/Java/HW/src/ex1 main base 3.12.7
# ls

Directory: E:\CODE\Java\HW\src\ex1

Mode                LastWriteTime         Length Name
----                -
-a----             3/2/2025   22:35           448 My.class
-a----             2/28/2025   10:35           533 My.java
-a----             2/28/2025   11:01           957 p10_l1.class
-a----             2/28/2025   11:01          1577 p10_l1.java
-a----             2/28/2025   11:18           558 p11_l2.class
-a----             2/28/2025   11:18           790 p11_l2.java
-a----             2/28/2025   11:21           784 p13_e1.class
-a----             2/27/2025   23:10           509 p13_e1.java
-a----             3/2/2025   22:35           518 you.class
```

2. 因为`you`类中没有`main`方法, `java`以`main`方法作为程序的入口

```
package ex1;

public class My {
    public static void main(String[] args) {
        System.out.println(you.info);
    }
}

class you {
    static String info = "同学们好!";
    public static void main(String[] args) {
        System.out.println(info);
    }
}
```

```
Windows PowerShell
@29701 on E:/CODE/Java/HW/src/main base 3.12.7
# java ex1.you
Error: Main method not found in class ex1.you
please define the main method as:
    public static void main(String[] args)
or a JavaFX application class must extend javafx.application.Application
@29701 on E:/CODE/Java/HW/src/main base 3.12.7
#
```

```
public class My {
    public static void main(String[] args) {
        System.out.println(you.info);
    }
}

class you {
    static String info = "同学们好!";
    public static void main(String[] args) {
        System.out.println(info);
    }
}
```

3. 重新编译后执行 java you 成功输出 同学们好！

```
package ex1;

public class My {
    public static void main(String[] args) {
        System.out.println(you.info);
    }
}

class you {
    static String info = "同学们好!";
    public static void main(String[] args) {
        System.out.println(info);
    }
}
```

```
@29701 on E:/CODE/J...
# java ex1.you
Error: Main method not
public static void
or a JavaFX applicatio
@29701 on E:/CODE/J...
# javac .\ex1\My.java
@29701 on E:/CODE/J...
# java ex1.you
同学们好！
@29701 on E:/CODE/J...
#
```

4. 将源程序名改为 you.java 之后无法通过编译, 报错如下: error: class My is public, should be declared in a file named My.java

所以可以知道如果一个类被声明为 public (例如 public class My), 那么 文件名必须与 public 类的名称完全一致, 即文件名必须是 My.java

```
@p10_l1.java @p11_l2.java @p13_e1.java @you.java
1 package ex1;
2
3 public class My {
4     public static void main(String[] args) {
5         System.out.println(you.info);
6     }
7 }
8
9 class you {
10     static String info = "同学们好!";
11     public static void main(String[] args) {
12         System.out.println(info);
13     }
14 }
15
16 /*
17 思考:
18 1. 会产生两个类文件, 分别叫My.class you.class
19 2. 因为you类中没有main方法, java以main方法作为程序的入口
20 3. 重新使用 javac you.java 编译之后 使用java you执行程序, 则
```

```
@29701 on E:/CODE/Java/HW/src %main base 3.12.7
# java ex1.you
Error: Main method not found in class ex1.you, please define the main method as:
public static void main(String[] args)
or a JavaFX application class must extend javafx.application.Application
@29701 on E:/CODE/Java/HW/src %main base 3.12.7
# javac .\ex1\My.java
@29701 on E:/CODE/Java/HW/src %main base 3.12.7
# java ex1.you
同学们好！
@29701 on E:/CODE/Java/HW/src %main base 3.12.7
# javac .\ex1\My.java
error: file not found: .\ex1\My.java
Usage: javac <options> <source files>
use --help for a list of possible options
@29701 on E:/CODE/Java/HW/src %main base 3.12.7
# javac .\ex1\you.java
.\ex1\you.java:3: error: class My is public, should be declared in a file named My.java
public class My {
^
1 error
@29701 on E:/CODE/Java/HW/src %main base 3.12.7
```

2. 第二章 page10 基础练习1

(1) 变量的定义与赋值

```
public class p10_l1 {
    public static void main(String[] args) {
        int a = 20;
        boolean b = true;
        double c = 3.14159;
        char d = 'a';
        System.out.println(a + "," + b + "," + c + "," + d);
    }
}
```

此时程序编译运行输出 20,true,3.14159,a, 四种类型在一个print里面输出, 说明print中的+起到了将所有元素的输出作为字符串拼接在一起的作用

思考

1. 在a的定义语句下一行增加 a = 123456789 后重新编译正常输出 123456789,true,3.14159,a, 所以可以将123456789赋值给a
2. 可以通过另外一个转义字符\ 加在\前面即可将\ 赋值给d, 即: d = '\\'; 重新编译运行后成功输出 123456789,true,3.14159,\

```
package ex1;

public class p10_l1 {
    public static void main(String[] args) {
        int a = 20;
        a = 123456789;
        // int a = Math.random();
        // int a = (int) Math.random();
        // int a = (int) (Math.random() * 100);
        // int a = (int) (Math.random() * 81) + 10;
        boolean b = true;
        double c = 3.14159;
        char d = 'a';
        d = '\\';
        System.out.println(a + "," + b + "," + c + "," + d);
    }
}
```

```
@ 29701 on @ E:/CODE/Java/HW/src
# java ex1.p10_l1
123456789,true,3.14159,a
@ 29701 on @ E:/CODE/Java/HW/src
# javac .\ex1\p10_l1.java
@ 29701 on @ E:/CODE/Java/HW/src
# java ex1.p10_l1
123456789,true,3.14159,\
@ 29701 on @ E:/CODE/Java/HW/src
#
```

(2) 理解强制转换

① 将a的赋值语句修改成 `int a = Math.random();` 后编译报错: `incompatible types: possible lossy conversion from double to int`. 原因应该是 `Math.random()` 方法会生成一个double类型的数据, 无法赋值给定义为int类型的a

```
package ex1;

public class p10_l1 {
    public static void main(String[] args) {
        int a = 20;
        a = 123456789;
        int a = Math.random();
        // int a = (int) Math.random();
        // int a = (int) (Math.random() * 100);
        // int a = (int) (Math.random() * 81) + 10;
        boolean b = true;
        double c = 3.14159;
        char d = 'a';
        d = '\\';
        System.out.println(a + "," + b + "," + c + "," + d);
    }
}
```

```
@ 29701 on @ E:/CODE/Java/HW/src \main ● base 3.12.7
# javac .\ex1\p10_l1.java
.\ex1\p10_l1.java:7: error: incompatible types: possible lossy conversion from double to int
    int a = Math.random();
    ^
1 error
@ 29701 on @ E:/CODE/Java/HW/src \main ● base 3.12.7
#
```

② 改为 `int a = (int) Math.random();` 之后重新编译, 运行五次之后得到的输出结果均为 `0,true,3.14159,\` 也就是说在这五次中a都是0. 原因应该是使用 `Math.random()` 方法生成浮点数之后使用 `(int)` 进行强制类型转换向下取整, 使得a总为零

```
package ex1;

public class p10_l1 {
    public static void main(String[] args) {
        int a = 20;
        a = 123456789;
        int a = Math.random();
        // int a = (int) Math.random();
        // int a = (int) (Math.random() * 100);
        // int a = (int) (Math.random() * 81) + 10;
        boolean b = true;
        double c = 3.14159;
        char d = 'a';
        d = '\\';
        System.out.println(a + "," + b + "," + c + "," + d);
    }
}
```

```
@ 29701 on @ E:/CODE/Java/HW/src \main ● base 3.12.7
# javac .\ex1\p10_l1.java
@ 29701 on @ E:/CODE/Java/HW/src \main ● base 3.12.7
# java ex1.p10_l1
0,true,3.14159,\
@ 29701 on @ E:/CODE/Java/HW/src \main ● base 3.12.7
# java ex1.p10_l1
0,true,3.14159,\
@ 29701 on @ E:/CODE/Java/HW/src \main ● base 3.12.7
# java ex1.p10_l1
0,true,3.14159,\
@ 29701 on @ E:/CODE/Java/HW/src \main ● base 3.12.7
# java ex1.p10_l1
0,true,3.14159,\
@ 29701 on @ E:/CODE/Java/HW/src \main ● base 3.12.7
# java ex1.p10_l1
0,true,3.14159,\
@ 29701 on @ E:/CODE/Java/HW/src \main ● base 3.12.7
#
```

③ 改为 `int a = (int) (Math.random() * 100);` 之后多次运行, 发现输出a的值大致分布在0~100之间. 查询文档可知 `Math.random()` 方法产生的数据范围为[0,1)的double类型数据, `Math.random() * 100` 得到的数据范围则会是[0, 100). 然后经过强制类型转换为int后a为[0,99]的整数.

```
package ex1;

public class p10_l1 {
    public static void main(String[] args) {
        // int a = 20;
        // a = 123456789;
        // int a = Math.random();
        // int a = (int) Math.random();
        // int a = (int) (Math.random() * 100);
        // int a = (int) (Math.random() * 81) + 10;
        boolean b = true;
        double c = 3.14159;
        char d = 'a';
        d = '\\';
        System.out.println(a + ", " + b + ", " + c + ", " + d);
    }
}

/*
思考
1).
1. 成功将123456789赋给a
2. 使用另一个反斜杠转义字符即可将'\'赋值给d
2).
1. 尝试编译报错如下: incompatible types: possible lossy conversion from double to int
2. 替换为强制转换为int类型重新编译多次运行之后发现a一直是0, 可能是该方法有必要参数尚未填写, 在[0,1)的范围上乘以100得到[0,100)的范围, 再强制转换为int类型就是[0,81)的范围, 再加上10得到[10,90)的范围
3).
*/
```

思考

`Math.random()` 方法产生的是 $[0,1)$ 的随机浮点数, 想要得到 $[10,90]$ 的范围首先左边界无法通过乘法来得到, 所以在最外层需要加上 $+10$.

此时我们需要生成的范围就是 $[0,80]$, 根据左闭右开的原则强制转换为`int`类型就是 $[0,81)$.

$[0,1) \rightarrow [0,81)$, 所以再需要再 `random()` 生成的数上载乘以81

所以最终是 `int a = (int) (Math.random() * 81) + 10;` 能得到 $[10,90]$ 的数据

```
package ex1;

public class p10_l1 {
    public static void main(String[] args) {
        // int a = 20;
        // a = 123456789;
        // int a = Math.random();
        // int a = (int) Math.random();
        // int a = (int) (Math.random() * 100);
        // int a = (int) (Math.random() * 81) + 10;
        boolean b = true;
        double c = 3.14159;
        char d = 'a';
        d = '\\';
        System.out.println(a + ", " + b + ", " + c + ", " + d);
    }
}

/*
思考
1).
1. 成功将123456789赋给a
2. 使用另一个反斜杠转义字符即可将'\'赋值给d
2).
*/
```

第二章 page11 基础练习2: 典型运算符的使用

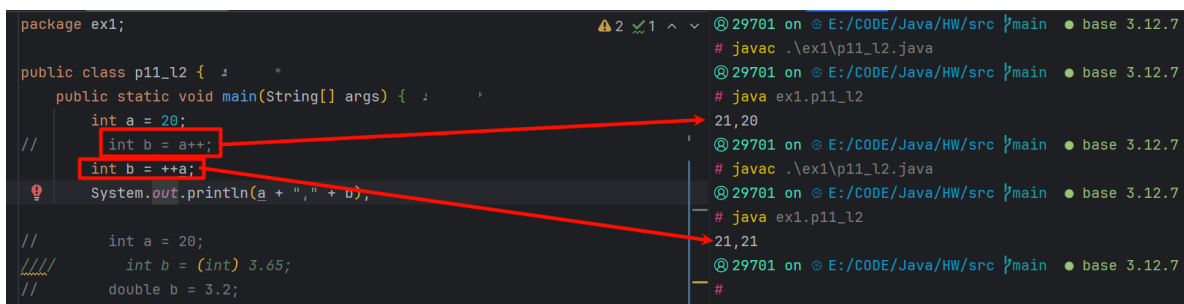
(1) 理解++运算符的位置差异

```
public class p11_l2 {
    public static void main(String[] args) {
        int a = 20;
        int b = a++;
        System.out.println(a + ", " + b);
    }
}
```

此时编译运行的输出为 21,20 , 将 `int b = a++` 改为 `int b = ++a` 后, 得到输出结果 21,21

`a++` 和 `++a` 的相同性在于: 他们都等价于 `a += 1`

不同性在于: `a++` 是执行完这条指令之后再将 `a+=1`, `++a` 是执行这条指令之前将 `a+=1`



```
package ex1;

public class p11_l2 {
    public static void main(String[] args) {
        int a = 20;
        // int b = a++;
        int b = ++a;
        System.out.println(a + ", " + b);
    }
}

// int a = 20;
// int b = (int) 3.65;
// double b = 3.2;
```

Terminal output:

```
@ 29701 on @ E:/CODE/Java/HW/src %main ● base 3.12.7
# javac .\ex1\p11_l2.java
@ 29701 on @ E:/CODE/Java/HW/src %main ● base 3.12.7
# java ex1.p11_l2
21,20
@ 29701 on @ E:/CODE/Java/HW/src %main ● base 3.12.7
# javac .\ex1\p11_l2.java
@ 29701 on @ E:/CODE/Java/HW/src %main ● base 3.12.7
# java ex1.p11_l2
21,21
@ 29701 on @ E:/CODE/Java/HW/src %main ● base 3.12.7
#
```

(2) 理解求余运算

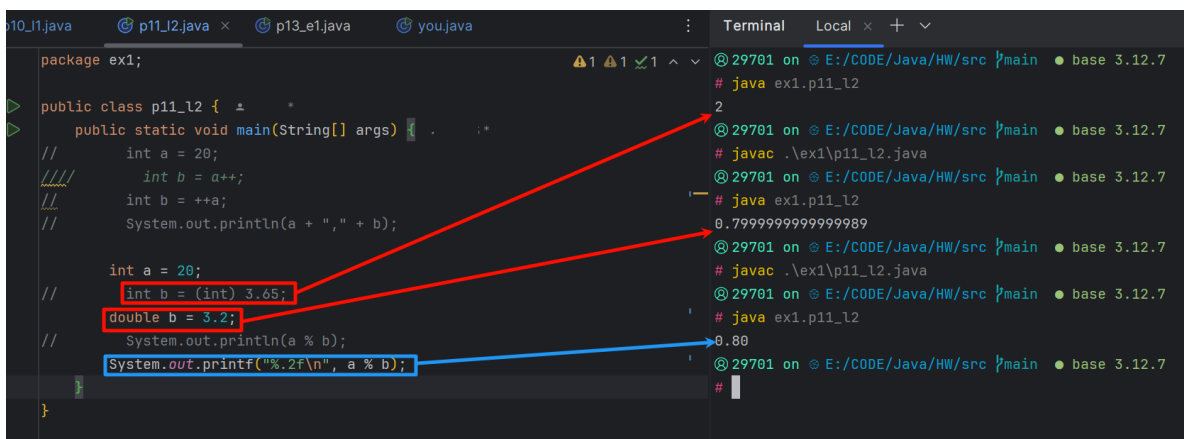
```
public class p11_l2 {
    public static void main(String[] args) {
        int a = 20;
        int b = (int) 3.65;
        System.out.println(a % b);
    }
}
```

该程序编译运行得到结果 2

将b的赋值修改 `double b = 3.2` 之后重新编译运行得到结果 0.7999999999999999

总结出混合类型运算的规律: 当运算中存在更高级的数据类型, 运算结果将和更高级的数据类型相同

通过数据语句改成 `System.out.printf("%.2f\n", a % b);` 得到输出结果 0.80



```
package ex1;

public class p11_l2 {
    public static void main(String[] args) {
        // int a = 20;
        // int b = a++;
        // int b = ++a;
        // System.out.println(a + ", " + b);

        int a = 20;
        // int b = (int) 3.65;
        double b = 3.2;
        // System.out.println(a % b);
        System.out.printf("%.2f\n", a % b);
    }
}
```

Terminal output:

```
@ 29701 on @ E:/CODE/Java/HW/src %main ● base 3.12.7
# java ex1.p11_l2
2
@ 29701 on @ E:/CODE/Java/HW/src %main ● base 3.12.7
# javac .\ex1\p11_l2.java
@ 29701 on @ E:/CODE/Java/HW/src %main ● base 3.12.7
# java ex1.p11_l2
0.7999999999999999
@ 29701 on @ E:/CODE/Java/HW/src %main ● base 3.12.7
# javac .\ex1\p11_l2.java
@ 29701 on @ E:/CODE/Java/HW/src %main ● base 3.12.7
# java ex1.p11_l2
0.80
@ 29701 on @ E:/CODE/Java/HW/src %main ● base 3.12.7
#
```

3. 第二章 page13 编程练习1

输入一个圆柱体的高和底面半径, 求其体积, 输出结果精确到小数点后3位

```
import java.util.Scanner;

public class p13_e1 {
    public static void main(String[] args) {
        Scanner sc = new Scanner(System.in);

        System.out.println("请输入圆柱体的高");
        double h = sc.nextDouble();
        System.out.println("请输入圆柱体底面半径");
```

```

double r = sc.nextDouble();

double v = Math.PI * r * r * h;

System.out.println("圆柱体的体积是");
System.out.printf("%.3f",v);

sc.close();
}
}

```

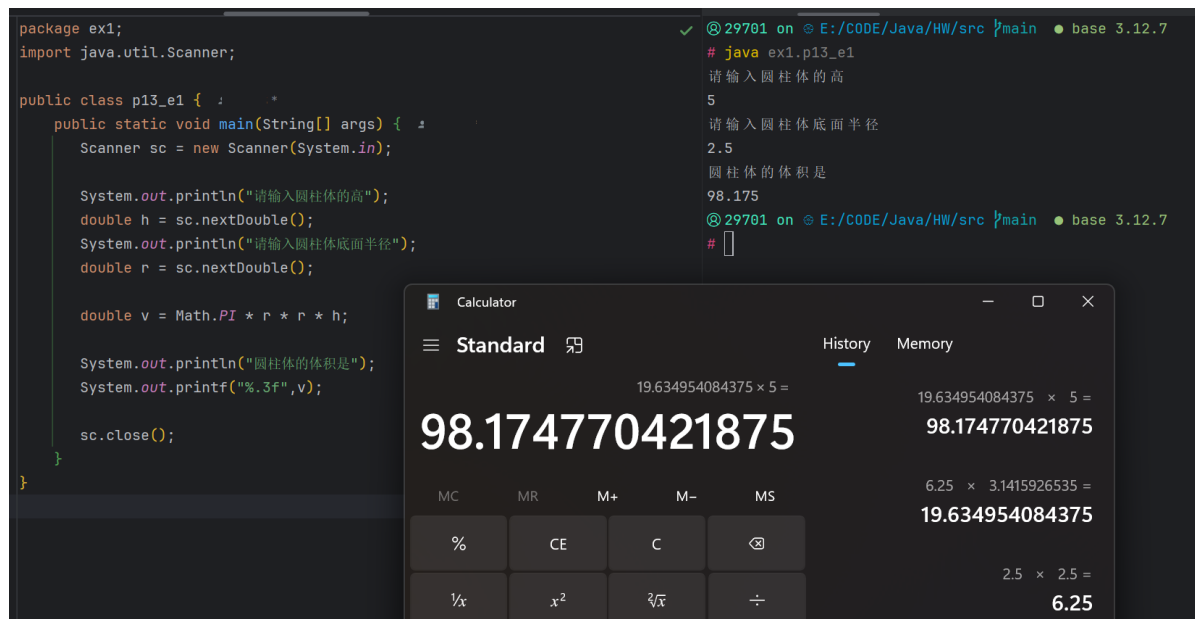
编译运行, 设定高为5, 底面半径为2.5, 得到输出:

```

请输入圆柱体的高
5
请输入圆柱体底面半径
2.5
圆柱体的体积是
98.175

```

经过计算器验算结果正确



五、实验体会

遇到的问题:

1. 数据类型转换错误

`Math.random()` 返回 `double`, 不能直接赋值给 `int`, 强制转换 `(int) Math.random()` 后总为0. 调整为 `(int) (Math.random() * 100)` 后, 成功生成0~99的随机整数.

另外的总结, 数据类型转换, 从高级到低级(比如`double`到`int`)就必须强制类型转换, 从低级到高级(如`byte`到`int`)就不需要手动进行强制转换, 而是被自动地隐式转换了.

2. ++ 运算符位置差异

`a++` 先赋值后递增, `++a` 先递增后赋值, 在循环和逻辑判断中需注意区别.

3. 求余运算的浮点误差

`a % b` 在 `b` 为浮点数时可能出现精度误差, 使用 `System.out.printf("%.2f", a % b);` 格式化输出可避免问题.

总而言之, 本次实验加深了我对Java数据类型, 运算符, 输入输出的理解, 并通过调试解决了类型转换, 浮点误差, 格式控制等问题, 体会到细节对程序正确性的重要性.