习题1

- 以下表达式是否能够通过编译？如果能够通过，表达式的类型是什么？表达式的值是什么？
  - 1 == True
  - 1 == true
  - 0 == false
  - 2 + "ab"
  - 2.3 + "ab"
  - 2 + 'a'
  - 2 * "ab"
  - 2 * 'a'
  - 1 + 1.0
  - 1/3
  - 1.0/3

习题1

- ~~1 == True~~
- ~~1 == true~~
- ~~0 == false~~
- 2 + “ab” -> String “2ab”
- 2.3 + “ab” -> String “2.3ab”
- 2 + ‘a’ -> int 99
- ~~2 * “ab”~~
- 2 * ‘a’ -> int 194

习题1

- 1 + 1.0 -> double 2.0
- 1 / 3 -> int 0
- 1.0 / 3 -> double 0.3333333333333333

习题2

- 假设 int a = 2147483647; （即，Integer.MAX_VALUE）.以下语句的输出值是什么？

```
System.out.println(a);
System.out.println(a + 1);
System.out.println(2 - a);
System.out.println(-2 - a);
System.out.println(2 * a);
System.out.println(4 * a);
```

习题2

- int a = 2147483647;

```
System.out.println(a);          ->   2147483647

System.out.println(a + 1);      ->   -2147483648

System.out.println(2 - a);      ->   -2147483645

System.out.println(-2 - a);     ->   2147483647

System.out.println(2 * a);      ->   -2

System.out.println(4 * a);      ->   -4
```

包装类

周云晓

2017.3.16

内容

- 数据
- 方法
- 装箱与拆箱

包装类

- 基本数据类型

基本数据类型	二进制位数
char	16bits
byte	8bits
short	16bits
int	32bits
long	64bits
float	32bits
double	64bits
boolean	--

包装类

- 为什么需要包装类？
  - Java语言是一个面向对象的语言，但是Java的基本数据类型却不是面向对象的
  - 在实际使用中经常需要将基本数据转化成对象，便于操作

包装类

- 基本数据类型 & 包装类

基本数据类型	包装类
byte	Byte
short	Short
int	Integer
long	Long
char	Character
float	Float
double	Double
boolean	Boolean

包装类

- 用途：
  - 将基本值“包装”到对象中，方便涉及到对象的操作
  - 包含每种基本数据类型的相关属性（最大值、最小值等），及相关的操作方法（基本值与String对象间相互转换等）

内容

- 数据
- 方法
- 装箱与拆箱

数据

- TYPE
  - 包装类所代表的基本数据类型

```
System.out.println(Byte.TYPE);
System.out.println(Short.TYPE);
System.out.println(Integer.TYPE);
System.out.println(Long.TYPE);
System.out.println(Double.TYPE);
System.out.println(Float.TYPE);
System.out.println(Character.TYPE);
System.out.println(Boolean.TYPE);
```



```
byte
short
int
long
double
float
char
boolean
```

数据 -- 2 & 3

- MIN_VALUE & MAX_VALUE
  - 数值类型包装类的基本类型的取值范围

```
System.out.println(Byte.MIN_VALUE);                    -128
System.out.println(Byte.MAX_VALUE);                    127
System.out.println(Short.MIN_VALUE);                   -32768
System.out.println(Short.MAX_VALUE);                   32767
System.out.println(Integer.MIN_VALUE);                 -2147483648
System.out.println(Integer.MAX_VALUE);                 2147483647
System.out.println(Long.MIN_VALUE);                    -9223372036854775808
System.out.println(Long.MAX_VALUE);                    9223372036854775807
System.out.println(Float.MIN_VALUE);                   1.4E-45
System.out.println(Float.MAX_VALUE);                   3.4028235E38
System.out.println(Double.MIN_VALUE);                  4.9E-324
System.out.println(Double.MAX_VALUE);                  1.7976931348623157E308
```

数据 -- 4

- SIZE
  - 包装类的基本类型的位数

```
System.out.println(Byte.SIZE);
System.out.println(Short.SIZE);
System.out.println(Integer.SIZE);
System.out.println(Long.SIZE);
System.out.println(Float.SIZE);
System.out.println(Double.SIZE);
System.out.println(Character.SIZE);
```

→

```
8
16
32
64
32
64
16
```

内容

- 数据
- 方法
- 装箱与拆箱

方法

- 构造函数
  - 带有基本值参数并创建包装类对象
    - E.g., Integer obj1 = **new** Integer(123);

      Double obj2 = **new** Double(1.2);
  - 带有字符串参数并创建包装类对象
    - E.g., Integer obj1 = **new** Integer(“123”);

      Double obj2 = **new** Double(“1.2”);

方法 -- 2

- **parseInt** -- 数字字符串转换为int数值
  - public static int parseInt(String s)
    - int i = Integer.parseInt(“123”);
  - public static int parseInt(String s, int radix)
    - 将字符串按照参数radix指定的进制转换为int数值
    - int n = Integer.parseInt(“12”,10); -> 12
    - int n = Integer.parseInt(“12”,16); -> 18

方法 -- 2

- 命令行参数args[]
  - 运行时： java ClassName para1 para2 …
  - 参数间空格隔开
  - String类型，在程序中可以根据需要将其转换为int, double等类型
    - ~~int i= (int)args[0];~~
    - int i = Integer.parseInt(args[0]);

方法 -- 3

- **toString** -- int类型转换为对应的String
  - public static String toString (int i)
    - int i = 12;
    - String s = Integer.toString(i);
  - public static String toString (int i, int radix)
    - 将int值转换为参数radix指定的进制的字符串
    - int i = 20;
    - String s = Integer.toString(i, 16); -> “14”

方法 -- 4

- **intValue --** 生成对象的基本值
  - public int intValue ()
    - Integer I = new Integer(123);
    - int i = I.intValue(); -> 123
    - double d = I.doubleValue(); -> 123.0
    - short t = I.shortValue(); -> 123
    - …

方法 -- 5

- **equals** -- 对同一类的两个对象进行比较
  - public boolean equals (Object o)
    - Integer obj1 = new Integer(123);
    - Integer obj2 = new Integer(123);
    - Boolean r = obj1.equlas(obj2);

内容

- 数据
- 方法
- 装箱与拆箱

装箱与拆箱

- 基本数据类型和对应的包装类可以相互转换
  - 装箱：由基本数据类型向对应的包装类转换
    - e.g., 把 int 包装成 Integer 类的对象
  - 拆箱：包装类向对应的基本数据类型转换
    - e.g., 把 Integer 类的对象重新简化为 int

手动装箱与拆箱

- int 和 Integer 的相互转换
  - 通过 Integer 类的构造方法将 int 装箱
  - 通过 Integer 类的 intValue 方法将 Integer 拆箱

```
public class Main {
    public static void main(String[] args) {
        int m = 500;
        Integer obj = new Integer(m);   // 手动装箱
        int n = obj.intValue();   // 手动拆箱
        System.out.println("n = " + n);

        Integer obj1 = new Integer( value: 500);
        System.out.println("obj 等价于 obj1？" + obj.equals(obj1));
    }
}
```

自动装箱与拆箱

- int 和 Integer 的相互转换
  - Java 1.5 之后可以自动拆箱装箱，即在进行基本数据类型和对应的包装类转换时，系统将自动进行

```
public class Main {
    public static void main(String[] args) {
        int m = 500;
        Integer obj = m;   // 自动装箱
        int n = obj;   // 自动拆箱
        System.out.println("n = " + n);

        Integer obj1 = 500;
        System.out.println("obj 等价于 obj1？" + obj.equals(obj1));
    }
}
```