

实验 02：数据类型

一、实验目的

- 掌握 Java 的基本数据类型和内置数据类型的使用。
- 熟悉字符串的处理，包括 `String` 类和 `StringBuffer` 类的使用。
- 理解基本数据类型的包装类及其应用场景。
- 掌握 `BigInteger` 类的使用，处理大整数运算。
- 熟悉 Java 中的日期和时间处理类，包括 `Date`、`SimpleDateFormat`、`Calendar` 以及 Java 8 新增的日期和时间类。

二、实验学时

2 学时

三、实验类型

验证性

四、实验需求

1、硬件

每人配备计算机 1 台，建议优先使用个人计算机开展实验。

2、软件

IntelliJ IDEA，以及 Java 运行所需要的相关基础环境。

3、网络

本地主机能够访问互联网和实验中心网络。

4、工具

无。

五、实验任务

1. 编写程序，演示基本数据类型的使用。
2. 使用 `String` 类和 `StringBuffer` 类进行字符串操作。
3. 使用基本数据类型的包装类进行数据转换和操作。
4. 使用 `BigInteger` 类进行大整数运算。
5. 使用 `Date`、`SimpleDateFormat`、`Calendar` 以及 Java 8 新增的日期和时间类进行日期和时间的处理。

六、实验内容及步骤

1. 基本数据类型的使用

步骤：

- 声明并初始化各种基本数据类型的变量。
- 打印这些变量的值。

示例代码：

Java

```
1 public class PrimitiveTypes {  
2     public static void main(String[] args) {  
3         // 声明并初始化基本数据类型变量  
4         byte b = 127;  
5         short s = 32767;  
6         int i = 2147483647;  
7         long l = 9223372036854775807L;  
8         float f = 3.14f;  
9         double d = 3.141592653589793;  
10        char c = 'A';  
11        boolean bool = true;  
12  
13        // 打印这些变量的值  
14        System.out.println("byte: " + b);  
15        System.out.println("short: " + s);  
16        System.out.println("int: " + i);  
17        System.out.println("long: " + l);  
18        System.out.println("float: " + f);  
19        System.out.println("double: " + d);  
20        System.out.println("char: " + c);  
21        System.out.println("boolean: " + bool);  
22    }  
23 }
```

The screenshot shows the IntelliJ IDEA interface with the following details:

- File Bar:** File Edit View Navigate Code Analyze Refactor Build Run VCS Window Help
- Project Bar:** ruoyi - PrimitiveTypes.java [ruoyi-admin] - IntelliJ IDEA
- Toolbars:** Standard, Editor, Navigation, Search, Git.
- Code Editor:** The code for `PrimitiveTypes.java` is displayed. The line `float f = 3.14f;` is highlighted in yellow.
- Run Tab:** Shows the run configuration for `PrimitiveTypes` with the command `C:\jdk1.8.0_291\bin\java.exe ...`. The output window displays the following results:

```
byte: 127  
short: 32767  
int: 2147483647  
long: 9223372036854775807  
float: 3.14  
double: 3.141592653589793  
char: A  
boolean: true
```
- Bottom Status Bar:** Shows the build status: "Build completed successfully in 3 s 474 ms (moments ago)". It also includes file encoding (CRLF), character set (UTF-8), and other system information.

2. 字符串处理

步骤：

- 使用 `String` 类进行字符串的创建、连接、比较等操作。
- 使用 `StringBuffer` 类进行字符串的追加、插入、删除等操作。

示例代码：

Java

```
1 public class StringHandling {
2     public static void main(String[] args) {
3         // 使用String类
4         String str1 = "Hello";
5         String str2 = "World";
6         String str3 = str1 + " " + str2; // 字符串连接
7         System.out.println("Concatenated String: " + str3);
8         System.out.println("Length of str3: " + str3.length());
9         System.out.println("Substring of str3: " + str3.substring(0,
10            5));
11        // 使用StringBuffer类
12        StringBuffer sb = new StringBuffer("Hello");
13        sb.append(" World"); // 追加字符串
14        sb.insert(5, " Java"); // 插入字符串
15        sb.delete(5, 10); // 删除字符串
16        System.out.println("StringBuffer: " + sb.toString());
17    }
18 }
```

The screenshot shows the IntelliJ IDEA interface with the following details:

- File Bar:** File, Edit, View, Navigate, Code, Analyze, Refactor, Build, Run, VCS, Window, Help.
- Project Bar:** forhis-interface-forward > ruoyi-admin > src > main > java > com > ruoyi > demo > StringHandling.java [ruoyi-admin] - IntelliJ IDEA
- Code Editor:** The file StringHandling.java contains code demonstrating string concatenation and manipulation using StringBuffer. The code is as follows:

```
String str1 = "Hello";
String str2 = "World";
String str3 = str1 + " " + str2; // 字符串连接
System.out.println("Concatenated String: " + str3);
System.out.println("Length of str3: " + str3.length());
System.out.println("Substring of str3: " + str3.substring(0, 5));

// 使用StringBuffer类
StringBuffer sb = new StringBuffer("Hello");
sb.append(" World"); // 追加字符串
sb.insert(5, str: " Java"); // 插入字符串
sb.delete(5, 10); // 删除字符串
System.out.println("StringBuffer: " + sb.toString());
```

- Run Tab:** Shows the output of the run command. The output window displays the following text:

```
C:\jdk1.8.0_291\bin\java.exe ...
Concatenated String: Hello World
Length of str3: 11
Substring of str3: Hello
StringBuffer: Hello World

Process finished with exit code 0
```

- Bottom Status Bar:** Build completed successfully in 3 s 623 ms (moments ago)
- Right Sidebar:** Maven, Database, Art, Maven Executor, Bean Validation.

3. 基本数据类型的包装类

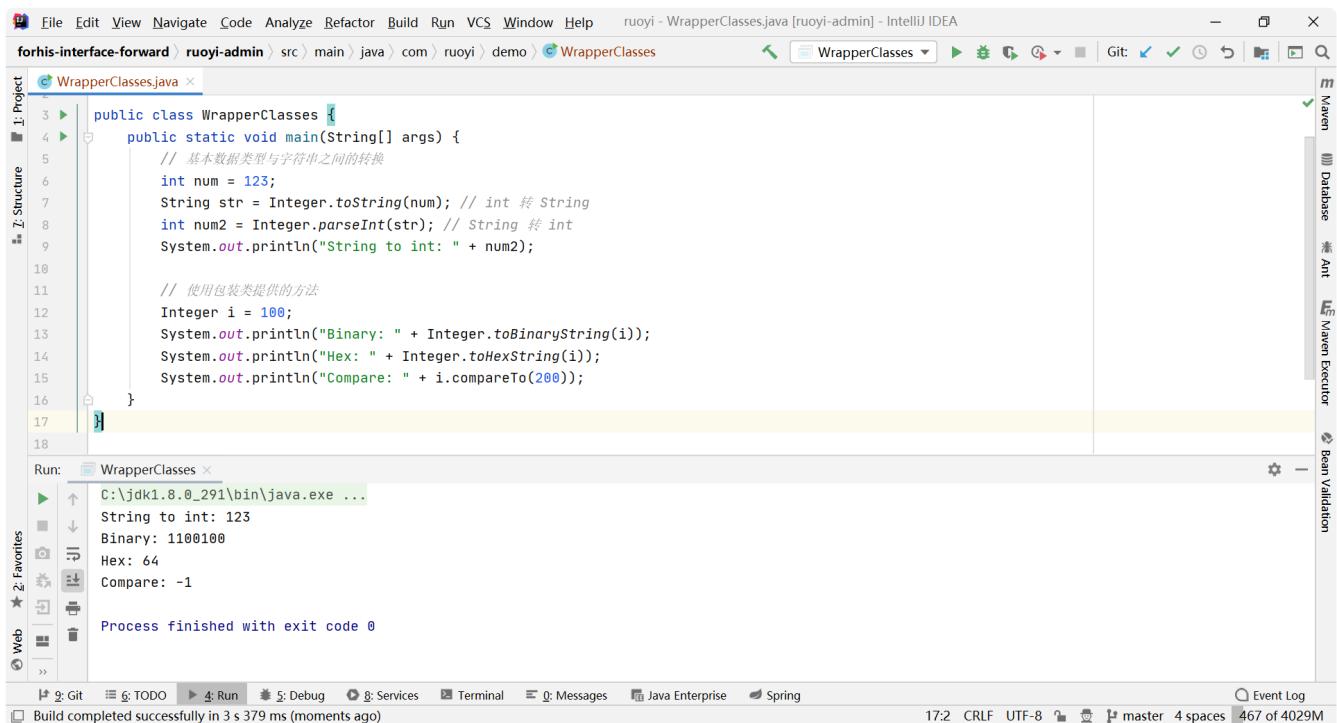
步骤：

- 使用包装类进行基本数据类型与字符串之间的转换。
- 使用包装类提供的方法进行数据操作。

示例代码：

Java

```
1 public class WrapperClasses {  
2     public static void main(String[] args) {  
3         // 基本数据类型与字符串之间的转换  
4         int num = 123;  
5         String str = Integer.toString(num); // int 转 String  
6         int num2 = Integer.parseInt(str); // String 转 int  
7         System.out.println("String to int: " + num2);  
8  
9         // 使用包装类提供的方法  
10        Integer i = 100;  
11        System.out.println("Binary: " + Integer.toBinaryString(i));  
12        System.out.println("Hex: " + Integer.toHexString(i));  
13        System.out.println("Compare: " + i.compareTo(200));  
14    }  
15 }
```



4. BigInteger 类的使用

步骤：

- 使用 `BigInteger` 类进行大整数的加减乘除运算。

示例代码：

Java

```
1 import java.math.BigInteger;
2
3 public class BigIntegerDemo {
4     public static void main(String[] args) {
5         BigInteger bigInt1 = new BigInteger("12345678901234567890");
6         BigInteger bigInt2 = new BigInteger("98765432109876543210");
7
8         // 加法
9         BigInteger sum = bigInt1.add(bigInt2);
10        System.out.println("Sum: " + sum);
11
12        // 减法
13        BigInteger difference = bigInt1.subtract(bigInt2);
14        System.out.println("Difference: " + difference);
15
16        // 乘法
17        BigInteger product = bigInt1.multiply(bigInt2);
18        System.out.println("Product: " + product);
19
20        // 除法
21        BigInteger quotient = bigInt2.divide(bigInt1);
22        System.out.println("Quotient: " + quotient);
23    }
24 }
```

The screenshot shows the IntelliJ IDEA interface with the Java code for `BigIntegerDemo.java`. The code demonstrates various arithmetic operations using the `BigInteger` class. In the bottom panel, the run output shows the results of the operations:

```
Sum: 1111111101111111100
Difference: -86419753208641975320
Product: 1219326311370217952237463801111263526900
Quotient: 8

Process finished with exit code 0
```

The IntelliJ IDEA interface includes toolbars, a navigation bar, and various tool windows on the right side.

5. 日期和时间类的使用

步骤：

- 使用 `Date` 类和 `SimpleDateFormat` 类进行日期和时间的格式化。
- 使用 `Calendar` 类进行日期和时间的操作。
- 使用 Java 8 新增的日期和时间类（如 `LocalDate`、`LocalTime`、`LocalDateTime`）进行日期和时间的处理。

示例代码：

Java

```
1 import java.text.SimpleDateFormat;
2 import java.util.Calendar;
3 import java.util.Date;
4 import java.time.LocalDate;
5 import java.time.LocalDateTime;
6 import java.time.format.DateTimeFormatter;
7
8
9 public class DateTimeDemo {
10     public static void main(String[] args) {
11         // 使用Date类和SimpleDateFormat类
12         Date date = new Date();
13         SimpleDateFormat sdf = new SimpleDateFormat("yyyy-MM-dd HH:mm:ss");
14         System.out.println("Formatted Date: " + sdf.format(date));
15
16         // 使用Calendar类
17         Calendar calendar = Calendar.getInstance();
18         calendar.setTime(date);
19         calendar.add(Calendar.DAY_OF_MONTH, 7); // 增加7天
20         System.out.println("Date after 7 days: " + sdf.format(calendar.getTime()));
21
22         // 使用Java 8新增的日期和时间类
23         LocalDate localDate = LocalDate.now();
24         LocalTime localTime = LocalTime.now();
25         LocalDateTime localDateTime = LocalDateTime.now();
26         DateTimeFormatter dtf = DateTimeFormatter.ofPattern("yyyy-MM-dd HH:mm:ss");
27
28         System.out.println("LocalDate: " + localDate);
29         System.out.println("LocalTime: " + localTime);
30         System.out.println("LocalDateTime: " + dtf.format(localDateTime));
31
32         // 获取当前日期
33         LocalDate today = LocalDate.now();
34         System.out.println("今天的日期是: " + today);
35
36         // 创建特定日期
37         LocalDate specificDate = LocalDate.of(2025, 1, 1);
38         System.out.println("特定日期是: " + specificDate);
```

```

39
40      // 日期加减
41      LocalDate tomorrow = today.plusDays(1);
42      LocalDate lastMonth = today.minusMonths(1);
43      System.out.println("明天是: " + tomorrow);
44      System.out.println("上个月同期是: " + lastMonth);
45
46      // 日期比较
47      if (today.isAfter(specificDate)) {
48          System.out.println("今天晚于特定日期");
49      }
50  }
51 }

```

The screenshot shows the IntelliJ IDEA interface with the following details:

- File Bar:** File, Edit, View, Navigate, Code, Analyze, Refactor, Build, Run, VCS, Window, Help.
- Project Bar:** ruoyi - DateTimeDemo.java [ruoyi-admin] - IntelliJ IDEA.
- Code Editor:** The file `DateTimeDemo.java` is open, containing Java code for date manipulation and comparison.
- Run Tab:** Shows the run configuration `DateTimeDemo` and its output. The output shows the current date, a specific date (2025-01-01), and the results of the date operations.
- Bottom Status Bar:** Shows the current time (5:26), file encoding (CRLF), and other system information.

```

// 创建特定日期
LocalDate specificDate = LocalDate.of( year: 2025, month: 1, dayOfMonth: 1);
System.out.println("特定日期是: " + specificDate);

// 日期加减
LocalDate tomorrow = today.plusDays(1);
LocalDate lastMonth = today.minusMonths(1);
System.out.println("明天是: " + tomorrow);
System.out.println("上个月同期是: " + lastMonth);

// 日期比较
if (today.isAfter(specificDate)) {
    System.out.println("今天晚于特定日期");
}

```

Run: DateTimeDemo

LocalDateTime: 2025-03-07 10:32:13
 今天的日期是: 2025-03-07
 特定日期是: 2025-01-01
 明天是: 2025-03-08
 上个月同期是: 2025-02-07
 今天晚于特定日期

Process finished with exit code 0

七、实验考核

本实验考核采用【实验随堂查】方式开展。

每个实验完成后，在实验课上通过现场演示的方式向实验指导教师进行汇报，并完成现场问答交流。

每个实验考核满分 100 分，其中实验成果汇报 60 分，现场提问交流 40 分。

实验考核流程：

- (1) 学生演示汇报实验内容的完成情况，实验指导老师现场打分。

- (2) 指导老师结合实验内容进行提问，每位学生提问 2-3 个问题，根据回答的情况现场打分。
- (3) 实验考核结束后，进行公布成绩。