

# Java String (String, printf, format)

## 1. String String

“\*\*String (Escape Sequence)\*\* String String String String (String , , String ) String String .  
Java String \ String .

### 1.1 String String

String	String	String String
\n	String	String String
\t	String	String (String 4~8String )
\"	String String	"String String: \"String\""
\'	String String	'I\'m fine'
\\	String String	"C:\\Users\\Dain"
\r	String String	String String String (String String )
\b	String	String String String (String String )
\f	String String	String String (String String )

### 1.2 String String

```
System.out.println("String\tString");
System.out.println("String\t25");
System.out.println("String String: \"String\"");
System.out.println("C:\\Program Files\\Java");
```

编译 运行 :

```
编译 运行
编译 25
编译 输出: "编译编译"
C:\Program Files\Java
```

## 2. 编译 编译 编译 编译

Java编译 编译 编译 编译 编译 编译 编译 `System.out.printf()` 编译 `String.format()` 编译 编译 .

### 2.1 编译 编译

```
System.out.printf("编译", 编译, 编译...);
String result = String.format("编译", 编译, 编译...);
```

### 2.2 编译 编译 编译

编译 编译	编译	编译 编译
<code>%s</code>	编译	<code>"编译: %s" → 编译 编译</code>
<code>%d</code>	编译 (10编译 )	<code>"编译: %d"</code>
<code>%f</code>	编译 (编译 编译 )	<code>"编译: %.2f"</code>
<code>%c</code>	编译	<code>"编译: %c"</code>
<code>%n</code>	编译 (OS编译 编译 )	<code>"编译 编译"</code>
<code>%g</code>	编译 (%) 编译	<code>"100%% 编译"</code>

### 2.3 编译/编译 编译

编译	编译
<code>%5d</code>	编译 5编译 , 编译 编译
<code>%-5d</code>	编译 5编译 , 编译 编译
<code>%05d</code>	编译 0编译 编译 (编译 5编译 )

<code>%.2f</code>	格式符 占位符 分隔符
<code>%6.2f</code>	格式符 宽度 精度 分隔符 占位符

## 2.4 练习 2.4

```
String name = "小明";
int age = 25;
double score = 93.756;

System.out.printf("姓名: %s, 年龄: %d\n", name, age);
System.out.printf("成绩: %.2f\n", score);
System.out.printf("总分: %d%%\n", 100);
```

运行结果：

```
姓名: 小明, 年龄: 25
成绩: 93.76
总分: 100%
```

## 3. 练习 3.1

格式符	转义符	输出结果
<code>\n</code> <code>\t</code> <code>\\</code>	<code>\n</code> , <code>\t</code> , <code>\\</code>	换行 制表符 反斜杠
<code>%d</code> <code>%.2f</code> <code>%s</code>	<code>%d</code> , <code>%.2f</code> , <code>%s</code>	整数 浮点数 字符串

## 4. 练习 4.1 (ANSI 转义符)

Java 使用 ANSI 转义符来设置文本颜色，ANSI 转义符的格式如下：

```
public class ColorExample {
    public static final String RED = "\u001B[31m";
    public static final String RESET = "\u001B[0m";
}
```

```
public static void main(String[] args) {  
    System.out.println(RED + "   重置." + RESET);  
}  
}
```

	ANSI
	\u001B[31m
	\u001B[32m
	\u001B[33m
	\u001B[34m
	\u001B[0m

## 5.    ? ? ? ? ?

Java 使用 `java.time.LocalDateTime` 和 `DateTimeFormatter` 来操作日期时间。 / 格式 日期 时间 。

### 5.1 DateTimeFormatter 使用

```
import java.time.LocalDateTime;  
import java.time.format.DateTimeFormatter;  
  
LocalDateTime now = LocalDateTime.now();  
DateTimeFormatter fmt = DateTimeFormatter.ofPattern("yyyy-MM-dd HH:mm:ss");  
System.out.println("当前时间: " + now.format(fmt));
```

### 5.2 日期格式

格式符	说明	示例
yyyy	年份	2025
MM	月份	05
dd	日期	22
HH	小时 (24小时制)	14

hh	HH (12時間)	02
mm		07
ss		59
a	AM / PM	AM / PM
E	曜日	Wed

## 5.3 日付フォーマッタ

```
DateTimeFormatter customFmt = DateTimeFormatter.ofPattern("yyyy-MM-dd (E) a hh:mm");
String formatted = now.format(customFmt);
System.out.println("日付フォーマッタ: " + formatted);
```

## 6. ログ

Javaには、`java.util.logging.Logger`というクラスがある。

### 6.1 Logger の使い方

```
import java.util.logging.*;

Logger logger = Logger.getLogger("MyLogger");

logger.info("Info ログ");
logger.warning("Warning ログ");
logger.severe("Severe ログ");
```

### 6.2 ログ出力のフォーマット

```
String name = "太郎";
int age = 25;
logger.info(String.format("名前: %s, 年齢: %d", name, age));
```

## 6.3 ?? ?? ??

??	??
SEVERE	??? ??
WARNING	??
INFO	?? ??
CONFIG	?? ?? ??
FINE	??? ??? ??

## 6.4 ?? ?? ???????

- ?? ?? ?? (?? /?? )
- ?? ??? ?? (?? ,?? ?? ,??? ?? ? )
- logging.properties ??? ?? ?? ??