

# 1. 杂记

- `printf` 函数中 `%n` 和 `\n` 都表示换行;
- 关于断言:
  - 断言的形式:
    - `asser condition[:expr]`, 例: `assert(name!=null):"变量name为空null"`;
  - `-ea` 选项使断言有效, 也可以使用 `-da` 选项使断言无效 (默认为无效);
  - 可以通过在 `-ea` 或 `-da` 后面指定包名来使一个包的断言有效或无效
    - `-da:com.test`
  - 要使一个包中的所有子包中的断言能够有效或无效, 在包名后加上三个点
    - `-da:com.test...`
- `spring` 中的 `Assert` 类并不是 `java` 断言机制的使用, 而是一种检查参数有效性的便捷手段, 运行时是有效的;
- 踩坑: 注意 `ArrayList` 中的 `remove(int)` 和 `remove(Object)` 的区别: 在 `ArrayList<Integer>` 情况下, 小心出错;
- `strictfp` 关键字

`strictfp` 关键字确保您将在每个平台上获得相同的结果, 如果在浮点变量中执行操作。不同平台的精度可能不同, 这就是为什么 `java` 编程语言提供了 `strictfp` 关键字, 它用于在每个平台上获得相同的结果。

- `strictfp` 关键字可以应用于方法, 类和接口

```
strictfp class A{//strictfp applied on class

strictfp interface M{//strictfp applied on interface

class B{
    strictfp void m(){}//strictfp applied on method
}
```

## 2. 泛型

### 2.1 泛型类

泛型类, 是在实例化类的时候指明泛型的具体类型;

### 2.2 泛型方法

泛型方法, 是在调用方法的时候指明泛型的具体类型;

```
/**
 * 泛型方法的基本介绍
 * @param tClass 传入的泛型实参
 * @return T 返回值为T类型
 * 说明:
 * 1) public 与 返回值中间<T>非常重要, 可以理解为声明此方法为泛型方法。
 * 2) 只有声明了<T>的方法才是泛型方法, 泛型类中的使用了泛型的成员方法并不是泛型方法。
 * 3) <T>表明该方法将使用泛型类型T, 此时才可以在方法中使用泛型类型T。
 * 4) 与泛型类的定义一样, 此处T可以随便写为任意标识, 常见的如T、E、K、V等形式的参数常用于表示泛型。
 */
public <T> T genericMethod(Class<T> tClass)throws InstantiationException ,
    IllegalAccessException{
    T instance = tClass.newInstance();
    return instance;
}
```

```
}
```

注：没有声明<T>的方法不是泛型方法；