week02

内存管理

JVM内存管理

栈
- 数据存储：方法调用参数、局部变量（基本类型、对象引用）
- 线程私有，每个线程独立栈，栈帧（Stack Frame）入栈/出栈
- 内存分配：固定大小（通过-Xss设置），自动分配
- 溢出：StackOverflowError（递归过深）
- 无垃圾回收

堆
- 数据存储：对象实例、数组
- 线程共享，分代管理（Young, Old, Permanent）
- 内存分配：动态扩展（-Xms初始堆，-Xmx最大堆）
- 溢出：OutOfMemoryError: Java heap space（对象过多）
- 垃圾回收：Major GC, Minor GC

方法区
- 数据存储：类信息、常量、静态变量、JIT代码
- 线程共享，JDK8后为Metaspace（本地内存）
- 内存分配：动态扩展（-XX:MaxMetaspaceSize限制）
- 溢出：OutOfMemoryError: Metaspace（动态类加载过多）
- 垃圾回收：卸载不再使用的类

Presented with xmind
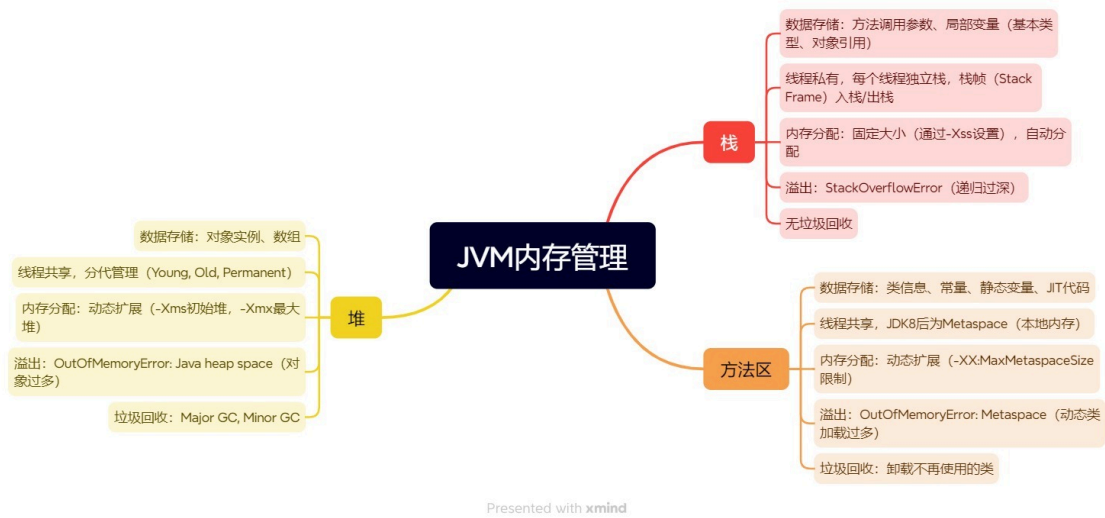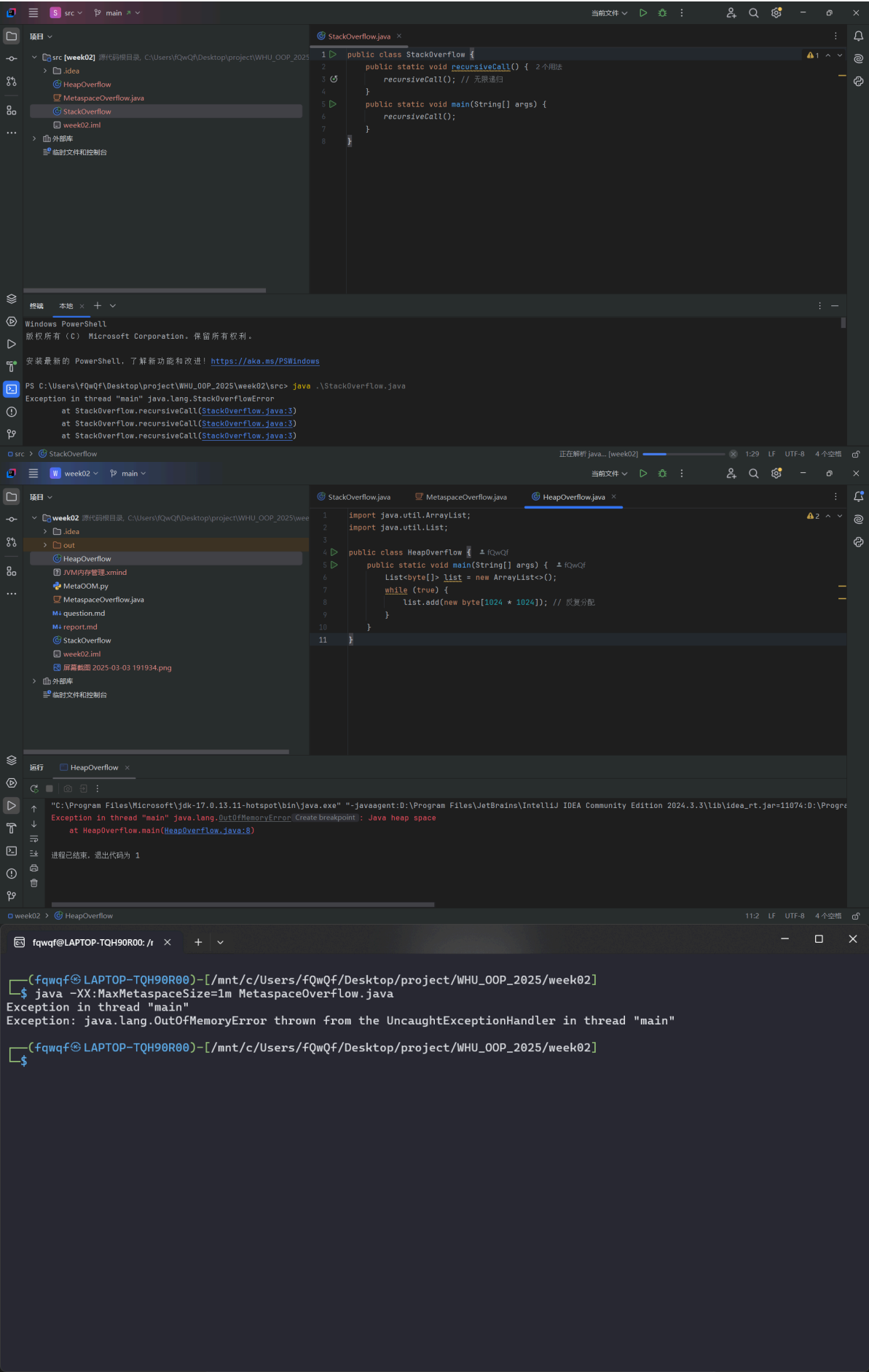
交互关系：方法调用时栈帧入栈，对象在堆中创建，类信息在方法区存储。垃圾回收主要针对堆，方法区卸载类较少发生。

运行结果

值得一提的是，为了使得方法区溢出，我写了一个python脚本，用来生成类。脚本如下：

```python
num_classes = 50000

with open("MetaspaceOverflow.java", "w", encoding="utf-8") as f:
    f.write("public class MetaspaceOverflow {\n")
    f.write("    public static void main(String[] args) {\n")
    f.write(f"        for (int i = 0; i < {num_classes}; i++) {{\n")
    f.write("            try {\n")
    f.write("                Class.forName(\"Class_\" + i);\n")
    f.write("            } catch (ClassNotFoundException e) {\n")
    f.write("                e.printStackTrace();\n")
    f.write("            }\n")
    f.write("        }\n")
    f.write("    }\n")
    f.write("}\n\n")

    for i in range(num_classes):
        f.write(f"class Class_{i} {{\n")
        f.write("    private static Object obj = new Object();\n")
        f.write("    public static void method() {}\n")
        f.write("}\n\n")
```

除此之外，我使用了 `-XX:MaxMetaspaceSize=1m` 来限制方法区的大小，方便溢出。