对07的回答

1.常见的Exception和Error：

Exception：

- NullpointException:调用的对象未初始化
- IndexOutOfBoundException:要求调用的数组或集合目标超出了其有效范围
- ArithmeticException:算术运算错误
- IOException:进行输入输出时出现错误

Error：

- OutOfMemoryError:内存不足
- NoClassDefFoundError:无法找到要调用的类
- UnsupportedClassVersionError:JVM不着急成绩当前文件
- StackOverflowError：递归调用过深，导致调用堆栈溢出

当发生Exception时：一般使用try-catch语块来捕获异常并进行处理

而发生Error时则说明出现了较为严重的的错误，一般需要对程序逻辑进行修改。

2. checked与unchecked异常
    - checked类型的异常在编译阶段会被发现，打断编译过程，较典型的例子如下
        1. IOException：输入输出时发生的异常
        2. ParseException：错误的解析数据
    - unchecked类型的异常在运行时才会被发现，具体例子如下：
        1. NullPointerException：调用一个空的对象
        2. IndexOutOfBoundException:要求调用的数组或集合目标超出了其有效范围

3.
```
class InsufficientFundsException extends Exception {//一个子类，用于账户余额不足报错
    public InsufficientFundsException(String message) {
        super(message);
    }
}

class BankAccount {//创建银行账户父类
    private double balance;

    public BankAccount(double initialBalance) {//简单的给账户写入一个余额
        this.balance = initialBalance;
    }

    public double getBalance() {获取账户余额
        return balance;
    }

    public void withdraw(double amount) throws InsufficientFundsException {
        if (amount > balance) {//定义一个余额不足的报错
```

```
            throw new InsufficientFundsException("余额不足，无法取款。当前余额: "
+ balance);//自定义报错信息
        }//如果余额不足，报错后停止运行，不会改变余额（我还想了一会这是不是bug，有被自己蠢
到）
        balance -= amount;
    }
}

public class BankAccountExample {
    public static void main(String[] args) {
        BankAccount account = new BankAccount(Math.random()*200);
//调用随机函数生成随机的余额
        try {
            System.out.println("当前余额: " + account.getBalance());
            account.withdraw(150.0);
            System.out.println("取款成功。");
        } catch (InsufficientFundsException e) {
            System.err.println("错误: " + e.getMessage());
        }//若余额充足，则运行try部分，错误则运行catch部分

        System.out.println("程序结束");
    }
}
```

总共有两种可能，要么余额充足取款成功，要么余额不足报错，具体流程见上方注释