* 今天学习目标

* 能够掌握Runtime常见的方法

* 通过查看源码初步理解单例模式

* 手写单例

* 1 构造器私有

* 2 静态方法返回引用

* 3 静态变量构建这个对象

* getRuntime， freeMemory，totalMemory，maxMemory，availableProcessors

* exec（mspaint,notepad,write）

* 能够理解异常的概述

* Throwable ， Error（ OutOfMemoryError） ， Exception（ RuntimeException， IOException）

* RuntimeException

*InputMismatchException

* NullPointerException

* ArrayIndexOutOfBoundsException

* ArithmeticException

* ....

* 能够掌握异常的5个关键字try，catch，finally，throw,throws

* try{ }catch(小){}catch(大){}finally{}

* throw在方法里抛异常

* throws 在方法签名

* try return

* 4个面试题

```
try{
    int i=1;
    return i;
```

```
        // throw new RuntimeException();
        }catch(Exception e){

      }finally{
       //  i=10;
        sysou("");
    }
```

* 能够掌握Java的异常体系

* 能够掌握自定义异常
* 能够掌握日志框架log4j和log4j2

---

* 回顾
  * 常见API
    * Object
      * native（JNI），equals,toString,clone,finalize,hashcode,getClass,notify,notifyAll,wait
    * DecimalFormat
      double d=8888.66666; 0.1-->10% "##.##%"
    * BigInteger,BigDecimal
    * 日期常见的类
      * Date，Calendar,SimpleDateFormat
      * LocalDate,LocalTime,LocalDateTime,DateTimeFormatter
    * String,Scanner,Math,Random,Arrays ,Byte,Short,Integer,Long,Character,Double...

* 能够掌握Runtime常见的方法
    * 通过查看源码初步理解单例模式

```
// 1 构造器私有
// 2 提供一个静态的方法:getInstance,getRuntime
    //     返回一个单例对象
// 3 可以提前先准备好
public class Singleton {
    // 3 准备好这个对象（只加载一次）
    private static  Singleton singleton=new Singleton();
    // 1 构造器私有
    private Singleton() {

    }
    // 2 写一个静态方法
    public static Singleton getInstance() {
        return singleton;
    }
}

Singleton instance1 = Singleton.getInstance();
Singleton instance2 = Singleton.getInstance();
System.out.println(instance1==instance2);
结果:
true
```

* getRuntime， freeMemory，totalMemory，maxMemory，availableProcessors

* exec (mspaint,notepad,write)

```
public static void main(String[] args) {
//      Runtime.getRuntime();
//      int[] max=new int[100000000];
//      Runtime.getRuntime().gc();
        //获取可用内存
        long value = Runtime.getRuntime().freeMemory();
        System.out.println("可用内存为:"+value/1024/1024+"MB");
        //获取jvm内存总数量，该值会不断的变化
        long  totalMemory = Runtime.getRuntime().totalMemory();
        System.out.println("全部内存为:"+totalMemory/1024/1024+"MB");
        //获取jvm 可以最大使用的内存数量，如果没有被限制 返回 Long.MAX_VALUE;
```

```
        long maxMemory = Runtime.getRuntime().maxMemory();
        System.out.println("可用最大内存为:"+maxMemory/1024/1024+"MB");
        //获得jvm运行可用核数
        int v = Runtime.getRuntime().availableProcessors();
        System.out.println(v);
        // 执行系统的命令
        try {
            Runtime runtime = Runtime.getRuntime();
            //mspaint,notepad,write
            runtime.exec("notepad");
        } catch (Exception e) {
            e.printStackTrace();
        }
    }
结果:
可用内存为:295MB
全部内存为:299MB
可用最大内存为:4425MB
8
```

* 能够理解异常的概述

  * 案例

```
public static void main(String[] args) {
    Scanner input=new Scanner(System.in);
    try {
    System.out.println("输入年龄: ");
    int age=input.nextInt();//java.util.InputMismatchException:类型不匹配异常
    System.out.println(age);
    }catch (InputMismatchException e) {//类型不匹配异常
        System.err.println("类型不匹配");//error
        System.out.println("输入年龄: ");
        input=new Scanner(System.in);
        int age=input.nextInt();
        System.out.println(age);
    }
}
```

```
public static void main(String[] args) {
        Scanner input=new Scanner(System.in);
```

```java
        try {
        System.out.println("输入年龄: ");
        int age=input.nextInt();//java.util.InputMismatchException:类型不匹配异常
        System.out.println(age);
        }catch (InputMismatchException e) {//类型不匹配异常
            System.err.println("类型不匹配");//error
            System.out.println("输入年龄: ");
            input=new Scanner(System.in);
            int age=input.nextInt();
            System.out.println(age);
        }
    }
```

```java
    public static void main(String[] args) {
        int a=12;
        int b=0;
        int result=a/b;
        System.out.println(result);
    }
}
```

Problems Javadoc Declaration Search Console

```
<terminated> Test2 (10) [Java Application] C:\Program Files\Java\jre1.8.0_181\bin\javaw.exe (2018年8月23日 上午9:25:08)
Exception in thread "main" java.lang.ArithmeticException: / by zero
        at com.hx.exception.Test2.main(Test2.java:7)
```

```java
public static void main(String[] args) {
        int a=12;
        int b=0;
        int result=a/b;
        System.out.println(result);
    }
```

```java
    public static void main(String[] args) {
        try {
            int a=12;
            int b=0;
            int result=a/b;
            System.out.println(result);
        }catch (ArithmeticException e) {
            e.printStackTrace();
            System.out.println("被除数不能为零");
        }
    }
}
```

Problems Javadoc Declaration Search Console

```
<terminated> Test2 (10) [Java Application] C:\Program Files\Java\jre1.8.0_181\bin\javaw.exe (2018年8月23日 上午9:29:43)
java.lang.ArithmeticException: / by zero
被除数不能为零
        at com.hx.exception.Test2.main(Test2.java:8)
```

```java
public static void main(String[] args) {
        try {
            int a=12;
            int b=0;
            int result=a/b;
            System.out.println(result);
        }catch (ArithmeticException e) {
            e.printStackTrace();
            System.out.println("被除数不能为零");
        }
    }
```

```java
public static void main(String[] args) {
    Scanner input=new Scanner(System.in);
    try {
        System.out.println("a:");
        int a=input.nextInt();
        System.out.println("b:");
        int b=input.nextInt();
        int result=a/b;
        System.out.println(result);
    }catch (ArithmeticException e) {
        e.printStackTrace();
        System.out.println("被除数不能为零");
    }finally {
        System.out.println("程序结束...");
    }

}
```

无论异常是否被捕获到，都会执行

```java
public static void main(String[] args) {
        Scanner input=new Scanner(System.in);
        try {
            System.out.println("a:");
            int a=input.nextInt();
            System.out.println("b:");
            int b=input.nextInt();
            int result=a/b;
            System.out.println(result);
        }catch (ArithmeticException e) {
            e.printStackTrace();
            System.out.println("被除数不能为零");
        }finally {
            System.out.println("程序结束...");
        }

    }
```

* 能够掌握异常的5个关键字try，catch，finally，throw,throws

* Java异常是Java提供的一种识别及响应错误的一致性机制。

* try -- 用于监听。将要被监听的代码(可能抛出异常的代码)放在try语句块之内，当try语句块内发生异常时，异常就被抛出。

* catch -- 用于捕获异常。catch用来捕获try语句块中发生的异常。

* finally -- finally语句块总是会被执行。它主要用于回收在try块里打开的物力资源(如数据库连接、网络连接和磁盘文件)。只要有finally块，执行完成之后，才会回来执行try或者catch块中的return或者throw语句，如果finally中使用了return或者throw等终止方法的语句，则就不会跳回执行，直接停止。

* throw -- 用于抛出异常。

- throws -- 用在方法签名中，用于声明该方法可能抛出的异常

```java
    public int devide(int a,int b) {
        try {
            int result=a/b;
            return result;
        } catch (Exception e) {
            e.printStackTrace();
            return 2;
        }finally {
            System.out.println("devide....");
            return 3;
        }
    }

    public static void main(String[] args) {
        Test3 test3=new Test3();
        int result=test3.devide(2, 0);
        System.out.println("result:"+result);
        System.out.println("main....");
    }
```

只有finally块，执行完成之后，才会回来执行try或者catch块中的return或者throw语句，如果finally中使用了return或者throw等终止方法的语句，则就不会跳回执行，直接停止。

```java
public int devide(int a,int b) {
        try {
            int result=a/b;
            return result;
        } catch (Exception e) {
            e.printStackTrace();
            return 2;
        }finally {
            System.out.println("devide....");
        }
```

```
    }

    public static void main(String[] args) {
        Test3 test3=new Test3();
        int result=test3.devide(2, 1);
        System.out.println("result:"+result);
        System.out.println("main....");
    }
```

```
public int devide(int a,int b) {
    int result=0;
    try {
        result=a/b;
    } catch (Exception e) {
        e.printStackTrace();
        // 处理...
        throw new ArithmeticException("被除数不能为0");
    }finally {
        System.out.println("devide....");
    }
    return result;
}
```

抛异常

```
public int devide(int a,int b) {
        int result=0;
        try {
            result=a/b;
        } catch (Exception e) {
            e.printStackTrace();
            // 处理...
            throw new ArithmeticException("被除数不能为0");
        }finally {
            System.out.println("devide....");
        }
        return result;
    }

    public static void main(String[] args) {
        Test4 test3=new Test4();
```

```
        try {
        int result=test3.devide(2, 0);
        System.out.println("result:"+result);
        }catch (ArithmeticException e) {
            System.out.println("被除数不能为0");
        }
        System.out.println("main....");
    }
```

常见面试题：

```
* 面试题一
public static void main(String[] args){
    int result = test1();
    System.out.println(result);
}

public static int test1(){
    int i = 1;
    try{
        i++;
        System.out.println("try block, i = "+i);
    }catch(Exception e){
        i--;
        System.out.println("catch block i = "+i);
    }finally{
        i = 10;
        System.out.println("finally block i = "+i);
    }
    return i;
}
* 结果
 try block, i = 2
 finally block i =10
 10

 * 面试题二
public static void main(String[] args){
    int result = test2();
```

```
    System.out.println(result);
}
public static int test2(){
    int i = 1;
    try{
        i++;
        throw new Exception();
    }catch(Exception e){
        i--;
        System.out.println("catch block i = "+i);
    }finally{
        i = 10;
        System.out.println("finally block i = "+i);
    }
    return i;
}

答案:
  catch block i =1
  finally block i =10
  10

面试题三:
public static void main(String[] args){
    int result = test3();
    System.out.println(result);
}

public static int test3(){
    //try 语句块中有 return 语句时的整体执行顺序
    int i = 1;
    try{
        i++;
        System.out.println("try block, i = "+i);
        return i;
    }catch(Exception e){
        i ++;
        System.out.println("catch block i = "+i);
        return i;
    }finally{
```

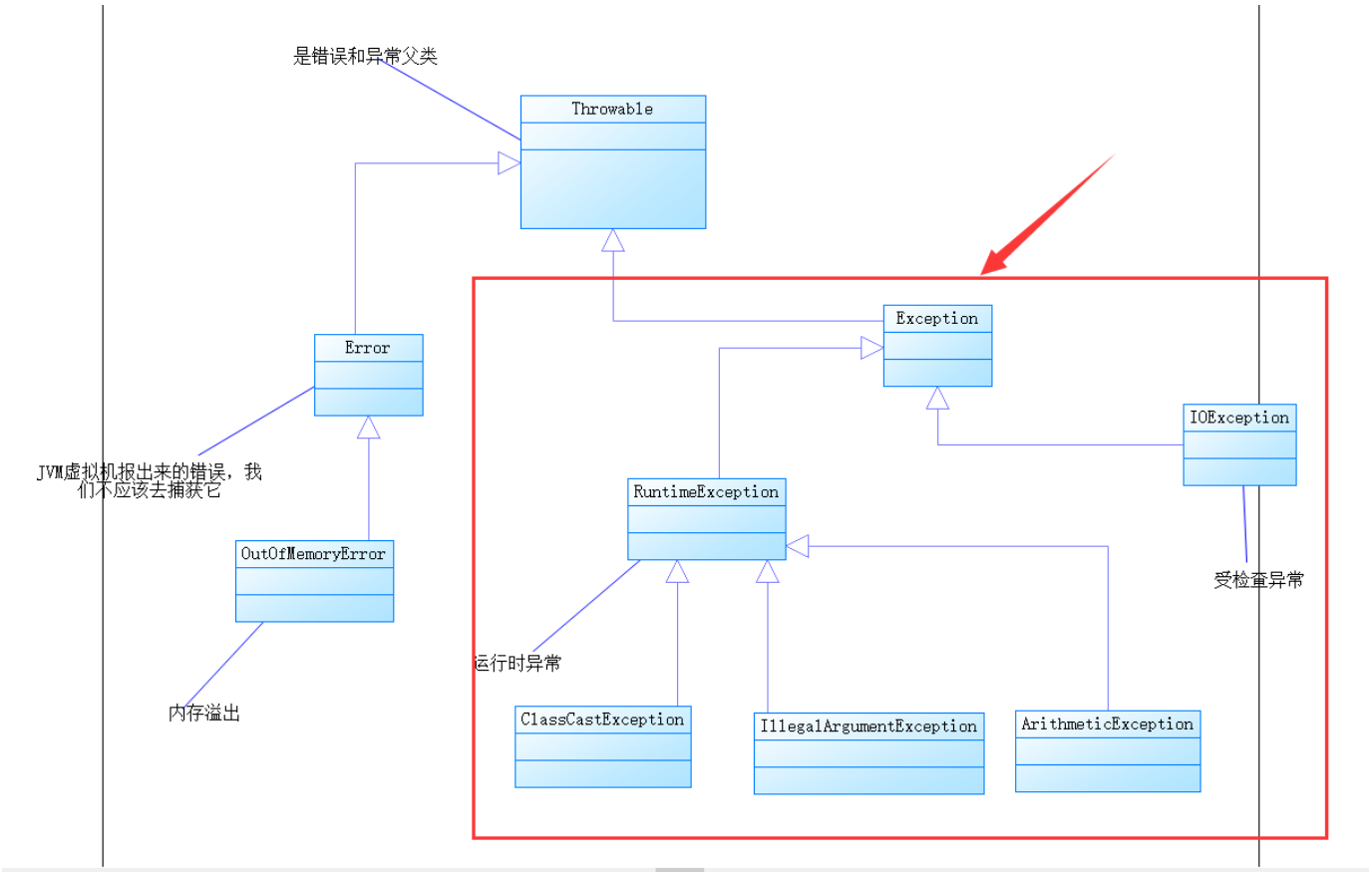
```
        i = 10;
        System.out.println("finally block i = "+i);
    }
}


答案:
   try block, i =2
   finally block i =10
   2
* 备注:
你会发现在 return 语句返回之前，虚拟机会将待返回的值压入操作数栈，
等待返回，即使 finally 语句块对 i 进行了修改，
但是待返回的值已经确实的存在于操作数栈中了，所以不会影响程序返回结果。

 面试题四
 public static void main(String[] args){
    int result = test4();
    System.out.println(result);
}
 public static int test4(){
    //finally 语句块中有 return 语句
    int i = 1;
    try{
        i++;
        System.out.println("try block, i = "+i);
        return i;
    }catch(Exception e){
        i++;
        System.out.println("catch block i = "+i);
        return i;
    }finally{
        i++;
        System.out.println("finally block i = "+i);
        return i;
    }
}
结果:
try block, i =2
finally block i =3
```

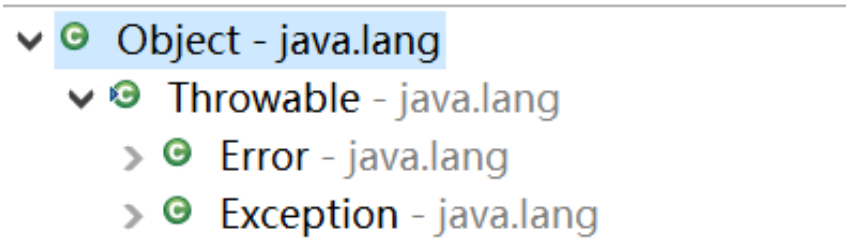
```
 3
* 参考链接
    * https://www.jianshu.com/p/49d2c3975c56
```

* 能够掌握Java的异常体系



* Throwable：

The {@code Throwable} class is the superclass of all errors and exceptions in the Java language

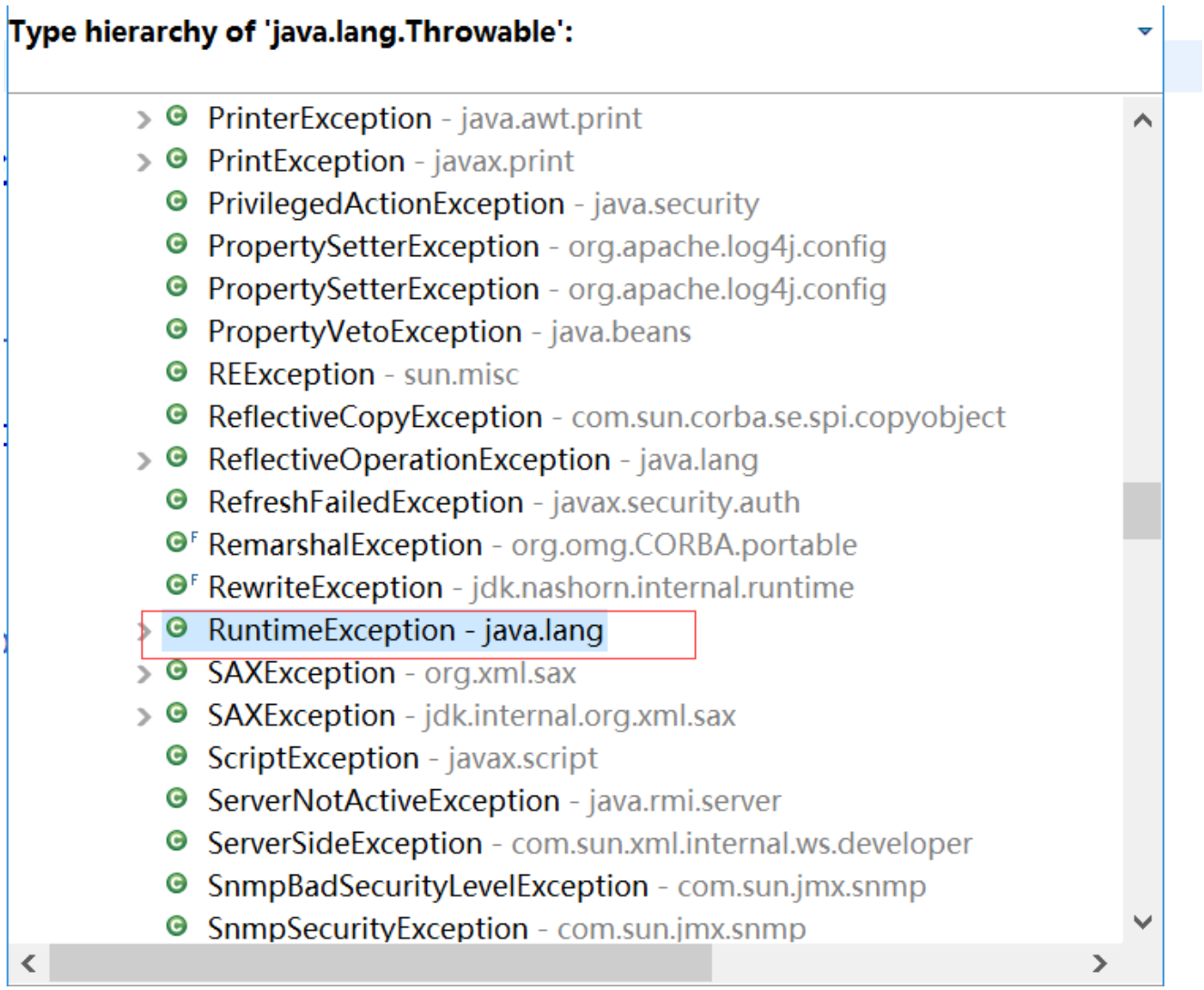


* Error

```
 An {@code Error} is a subclass of {@code Throwable}
 * that indicates serious problems that a reasonable applica
tion
 * should not try to catch
```

* Exception

```
The class {@code Exception} and its subclasses are a form of
 * {@code Throwable} that indicates conditions that a reason
able
 * application might want to catch
```



* RuntimeException

* 能够掌握自定义异常

* 能够掌握日志框架log4j和log4j2