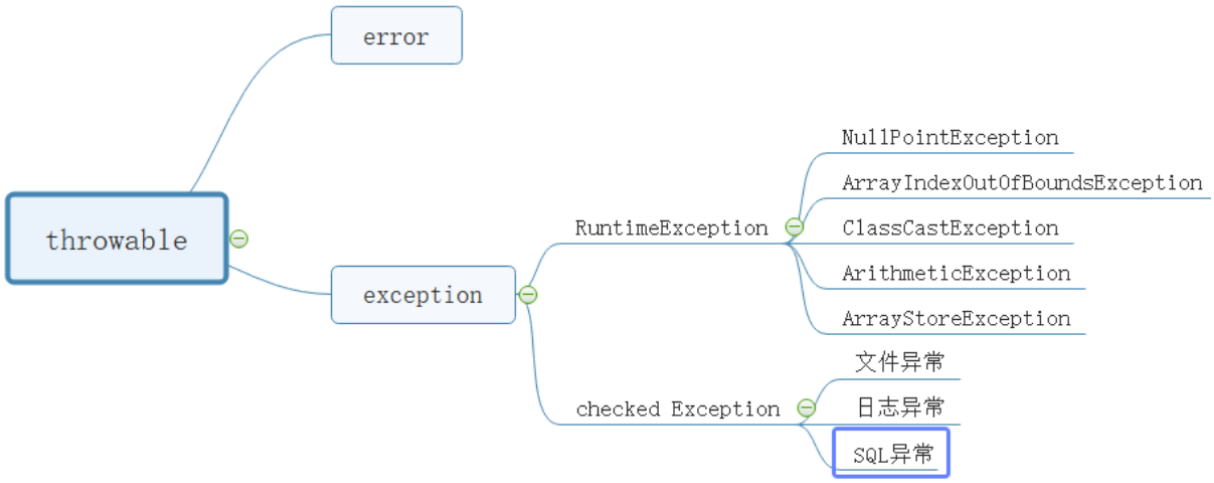
异常——答案

1. 异常体系问答

1）java中Throwable下的主要子类有（包括间接继承的子类），请利用思维导图的方式画出异常体系

答案：



2）简述Error和Exception区别

答案：

- Error表示错误情况，一般是程序出现了比较严重的问题，通常为虚拟机相关错误，如系统崩溃，内存不足，堆栈溢出等。编译器不会对这类错误进行检测，JAVA应用程序也不会对这类错误进行捕获，一旦这类错误发生，通常应用程序会被终止，仅靠应用程序本身无法恢复。
- Exception表示异常情况，大多数是可以通过特定的方式进行处理，并且处理完后，程序还可以继续往下正常运行。

3）简述运行时异常和编译时异常的区别

答案：

- 编译时异常继承自Exception，编译期间会主动检查这种异常，发现报错后要求我们处理。
- 运行时异常继承自RuntimeException，编译期间，不会检查这种异常，也不要处理，但是运行期间，可能抛出这种类型的异常。

4）写出5个常见的运行时异常（中文+英文）

答案：

- NullPointException 空指针异常
- ArrayIndexOutOfBoundException 数组下标越界异常
- ClassCastException 类型转换异常
- ArithmeticException 算术异常
- IIlegalArgumentException 非法参数异常
- ArrayStoreException 数组存储异常
- ……

2. 异常处理机制问答

1）简述使用catch(Exception e)的好处是

答案：

可以用来捕获try语句块中的所有类型的异常

2）简述当出现多个异常时，try-catch中catch子句的排列方式

答案：

子类异常在前，父类异常在后

3）简述throw 和 throws 的区别

【提示】

可从以下几个方面思考：

- 声明位置
- 抛出异常的个数
- 关键字作用

答案：

- 声明位置：throw关键字用在方法内部；throws关键字用在方法声明上
- 抛出异常的个数：throw关键字只能用于抛出一种异常；throws关键字可以抛出多个异常
- 关键字作用：throw关键字用来抛出方法或代码块中的异常，编译时异常和运行时异常都可以被抛出；throws关键字用来标识该方法可能抛出的异常列表。一个方法用throws标识了可能抛出的异常列表，调用该方法的方法中必须包含可处理异常的代码，否则也要在方法签名中用throws关键字声明相应的异常。

3. 异常编程

编写Test3_DivisionByZero类

【要求】

1. 编写方法：division()：要求执行10/0操作，并使用异常处理机制处理可能会产生的异常

   编写main()：调用division()

2. 修改division()：执行10/0不变，但不在方法中处理产生的异常，改将异常抛出

   修改main()：调用division()并处理其抛出的异常

答案：

```
// 第一题
package com.briup;
public class Test3_DivisionByZero {
    public static void main(String[] args) {
        DivisionByZero di = new DivisionByZero();
        di.division();
    }

    //在方法中捕获并处理异常
    public void division() {
        try {
            int a = 10 / 0;
        } catch (Exception e) {
            e.printStackTrace();
        }
    }
}
```

```
//第二题
package com.briup;
public class Test3_DivisionByZero {
    public static void main(String[] args) {
        DivisionByZero di = new DivisionByZero();
        //捕获并处理异常
        try {
            di.division();
```

```
        }catch(ArithmeticException e) {
            e.printStackTrace();
        }
    }

    //声明抛出异常
    public void division() throws ArithmeticException{
        int a = 10 / 0;
    }
}
```

4. finally问答

finally代码块是否一定会执行

答案：**不一定**

- 当程序进入try块之前就出现异常时，会直接结束，不会执行finally代码块中的代码
- 当程序在try块中强制退出时也不会去执行finally块中的代码，比如在try块中执行`System.exit(0)`方法。

5. finally代码分析

观察并分析以下代码，说出其输出结果

1)

```
package com.briup;

public class ReturnExceptionDemo {
    public static void methodA() throws Exception {
```

```
        try {
            System.out.println("进入方法A");
            throw new Exception("制造异常");
        } finally {
            System.out.println("用A方法的finally");
        }
    }

    public static int methodB() {
        try {
            System.out.println("进入方法B");
            return 1;
        } finally {
            System.out.println("用B方法的finally");
            return 2;
        }
    }

    public static void main(String[] args) {
        try {
            methodA();
        } catch (Exception e) {
            System.out.println(e.getMessage());
        }
        int i = methodB();
        System.out.println(i);
    }
}
```

1）输出结果

```
进入方法A
用A方法的finally
制造异常
进入方法B
用B方法的finally
1
```

2)

```
package com.briup;

public class Return1 {
    public static void main(String[] args) {
        int i = testReturn1();
        System.out.println(i);
    }

    private static int testReturn1() {
        int i = 1;
        try {
            i++;
            System.out.println("try:" + i);
            return i;
        } finally {
            i++;
            System.out.println("finally:" + i);
        }
    }
}
```

2）输出结果

```
try:2
finally:3
2
```

3)

```
package com.briup;

import java.util.ArrayList;
import java.util.List;
```

```
public class Return2 {
    public static void main(String[] args) {
        System.out.println(testReturn2());
    }

    private static List<Integer> testReturn2() {
        List<Integer> list = new ArrayList<>();
        try {
            list.add(1);
            System.out.println("try:" + list);
            return list;
        } finally {
            list.add(3);
            System.out.println("finally:" + list);
        }
    }
}
```

3) 输出结果

```
try:[1]
finally:[1, 3]
[1, 3]
```

4)

```
package com.briup;

public class Return3 {
    public static void main(String[] args) {
        System.out.println(testReturn3());
    }

    private static String testReturn3() {
        String a = "hello";
```

```
        try {
            a += "!";
            System.out.println("try:" + a);
            return a;
        } finally {
            a += "world";
            System.out.println("finally:" + a);
        }
    }
}
```

4）输出结果

```
try:hello!
finally:hello!world
hello!
```