

EXCEPTION HANDLING IN JAVA

Presented by:

Dr. Shivangi K. Surati

Assistant Professor,

Department of Computer Science and Engineering,

School of Technology,

Pandit Deendayal Energy University

Outline

- ❑ Error
- ❑ Exception
- ❑ Error vs exception
- ❑ Exception handling in Java
- ❑ Exception handling hierarchy
- ❑ Types of Exception handling
- ❑ Keywords
- ❑ Syntax
- ❑ Multiple catch clauses
- ❑ Nested try statements
- ❑ Methods generating exceptions
- ❑ Custom exceptions

Error

- ❑ “Error” is a **critical condition** that cannot be handled by the code of the program.
- ❑ An Error “indicates serious problems that a reasonable application should not try to catch.”
- ❑ Errors are the conditions which **cannot get recovered** by any handling techniques.
- ❑ It surely cause termination of the program abnormally.
- ❑ Errors belong to unchecked type and mostly occur at runtime.
- ❑ Some of the **examples** of errors are Out of memory error or a System crash error.

Error Example

```
class StackOverflow {
    public static void test(int i) {
        // No correct as base condition leads to non-stop
        recursion.
        if (i == 0)    return;
        else          test(i++);
    }
}

public class ErrorEg {
    public static void main(String[] args) {
        StackOverflow.test(5); // eg of StackOverflowError
    }
}
```

Output:
Exception in thread "main"
java.lang.StackOverflowError
at StackOverflow.test(ErrorEg.java:7)
at StackOverflow.test(ErrorEg.java:7)
at StackOverflow.test(ErrorEg.java:7)
at StackOverflow.test(ErrorEg.java:7)
at StackOverflow.test(ErrorEg.java:7)

Exception

- ❑ An Exception “indicates conditions that a reasonable application **might want to catch** (handled by the code of the program).”
- ❑ Exceptions are the **conditions that occur at runtime** and may cause the termination of program. But they are recoverable using try, catch and throw keywords.
- ❑ Exceptions are divided into two categories : checked and unchecked exceptions.
- ❑ Checked exceptions like IOException known to the compiler at compile time while unchecked exceptions like ArrayIndexOutOfBoundsException known to the compiler at runtime.
- ❑ It is mostly caused by the program written by the programmer (due to bad data provided by user).

Error Vs Exception

BASIS	ERROR	EXCEPTION
Basic	An error is caused due to lack of system resources.	An exception is caused because of the code.
Recovery	An error is irrecoverable.	An exception is recoverable.
Keywords	There is no means to handle an error by the program code.	Exceptions are handled using three keywords "try", "catch", and "throw".
Consequences	As the error is detected the program will terminated abnormally.	As an exception is detected, it is thrown and caught by the "throw" and "catch" keywords correspondingly.
Types	Errors are classified as unchecked type.	Exceptions are classified as checked or unchecked type.
Package	In Java, errors are defined "java.lang.Error" package.	In Java, an exceptions are defined in "java.lang.Exception".
Example	OutOfMemory, StackOverflow.	Checked Exceptions : NoSuchMethod, ClassNotFoundException. Unchecked Exceptions : NullPointerException, IndexOutOfBounds.

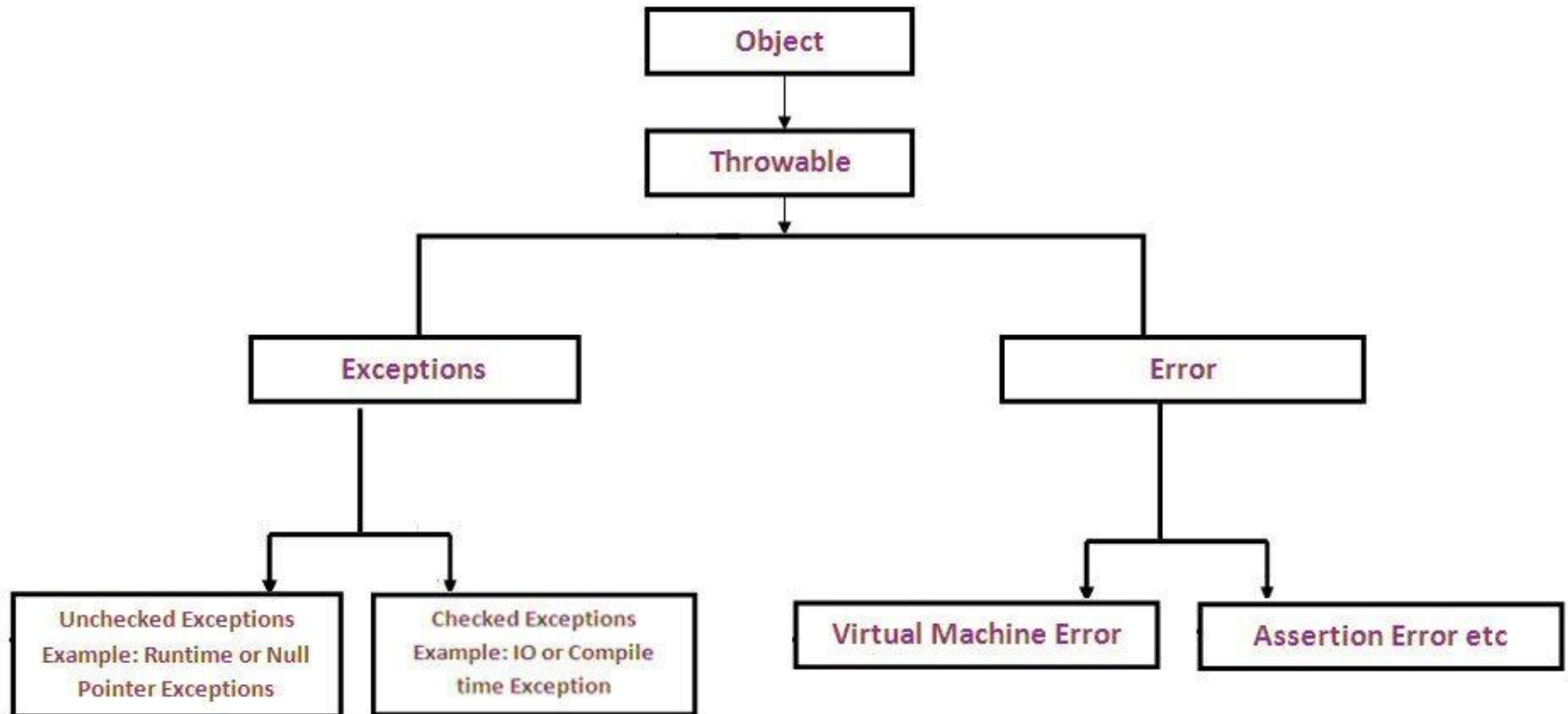
Exception handling in Java

- An Exception is an **unwanted event** that interrupts the normal flow of the program.
- System's exception handler
 - ▣ a system generated **error message** is shown to the user
 - ▣ program **execution gets terminated**
- The Exception Handling in Java is one of the powerful mechanism
 - ▣ to handle the runtime errors
 - ▣ While **maintaining normal flow of the application**
 - ▣ provide a **meaningful message (user friendly warning)** to the user about the issue rather than a system generated message-easy to understand
 - ▣ Let the users correct the error

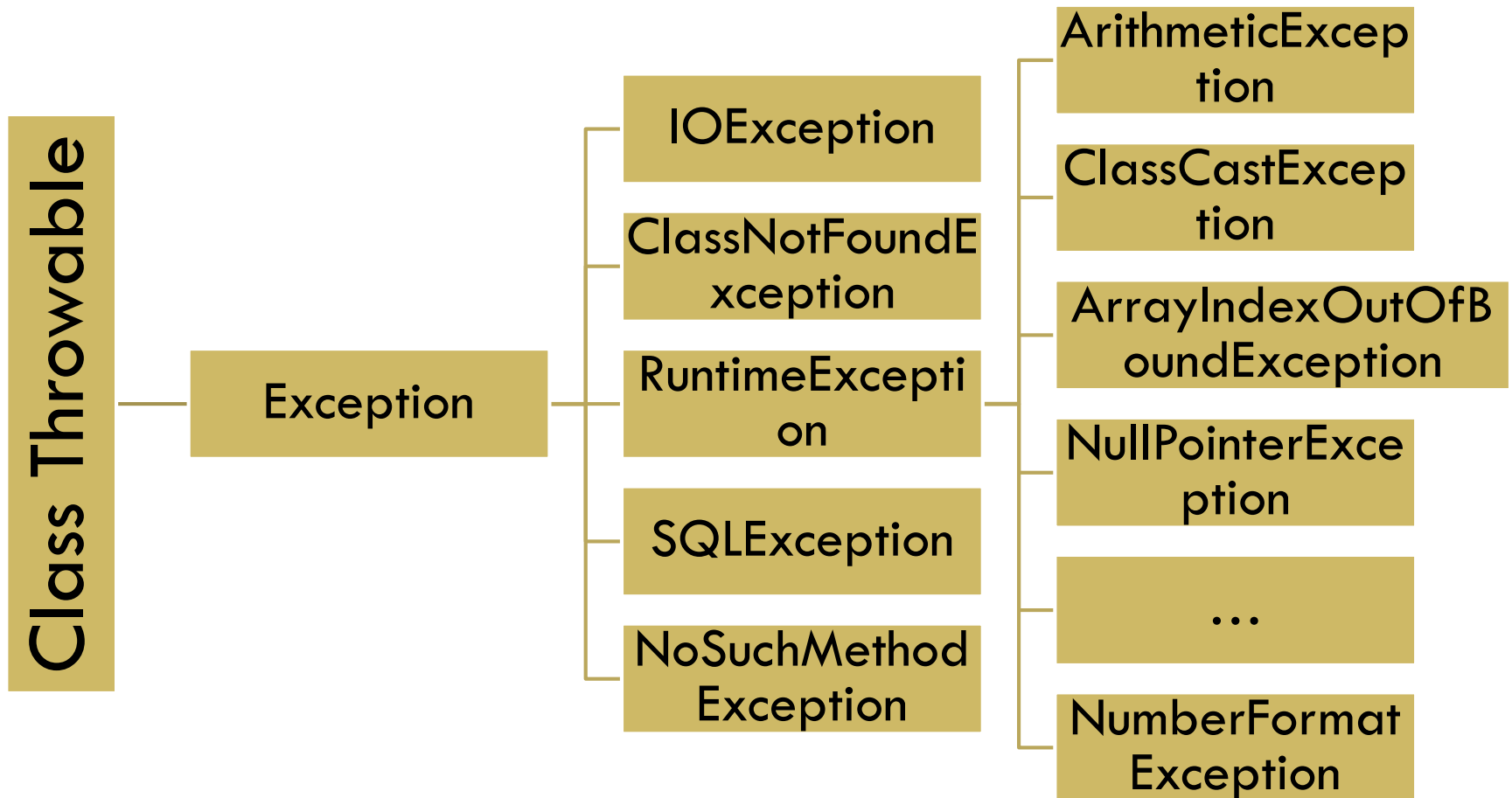
Exception handling- Advantages

- Ensures that the **flow of the program doesn't break when** an exception occurs.
- Custom exceptions can be defined in applications

Exception Handling Hierarchy



Exception Handling Hierarchy...



Types of Exceptions

- There are mainly two types of exceptions:
 - ▣ Checked and
 - ▣ Unchecked.
- Here, an error is considered as the unchecked exception.
- According to Oracle, there are three types of exceptions:
 - ▣ Checked Exception
 - ▣ Unchecked Exception
 - ▣ Error

Checked Exceptions

- The classes which directly inherit Throwable class except RuntimeException and Error are known as checked exceptions
- e.g. IOException, SQLException etc.
- Checked exceptions are **checked at compile-time.**

Unchecked Exception

- The classes which inherit RuntimeException are known as unchecked exceptions
- e.g. ArithmeticException, NullPointerException, ArrayIndexOutOfBoundsException etc.
- Unchecked exceptions are not checked at compile-time, but they are **checked at runtime.**

Keywords in Exception handling

Keyword	Description
try	The "try" keyword is used to specify a block where we should place exception code. The try block must be followed by either catch or finally. It means, we can't use try block alone.
catch	The "catch" block is used to handle the exception. It must be preceded by try block which means we can't use catch block alone. It can be followed by finally block later.
finally	The "finally" block is used to execute the important code of the program. It is executed whether an exception is handled or not.
throw	The "throw" keyword is used to throw an exception.
throws	The "throws" keyword is used to declare exceptions. It doesn't throw an exception. It specifies that there may occur an exception in the method. It is always used with method signature.

Exception handling in Java: Syntax

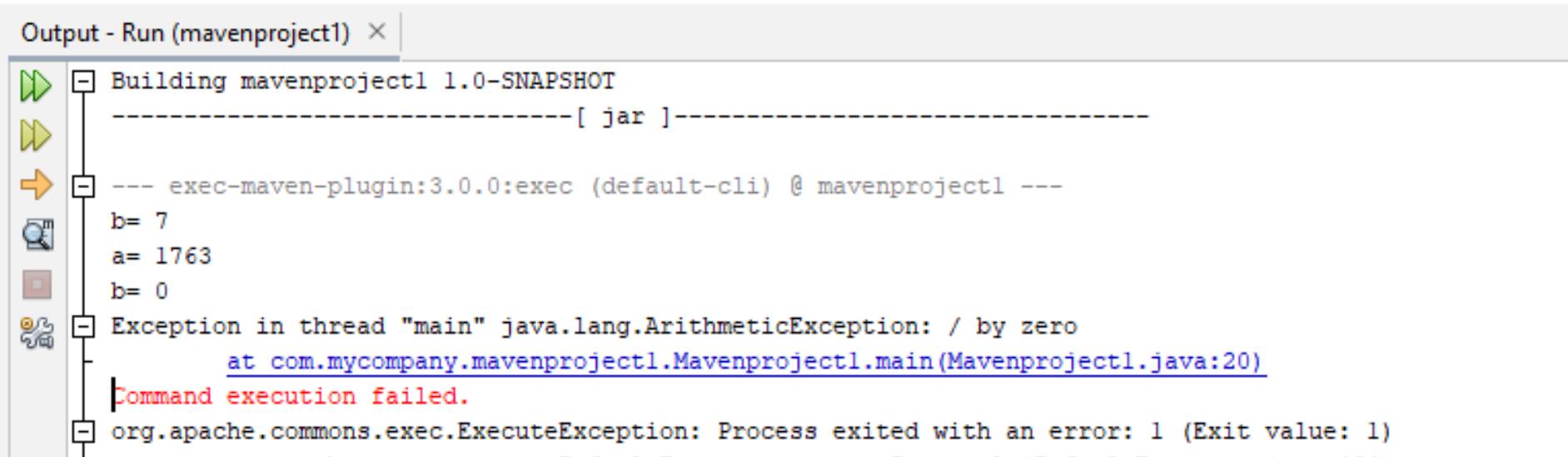
```
try {  
    // block of code  
}  
catch ( ExceptionType1 e) {  
    // Exception handling routine for ExceptionType1 (optional)  
}  
catch (ExceptionType2 e ) {  
    // Exception handling routine for ExceptionType2 (optional)  
}  
..  
catch (ExceptionType_n e) {  
    // Exception handling routine for ExceptionType_n (optional)  
}  
finally {  
    // Program code of exit (optional)    }
```

Exceptions

```
import java.util.Random;

public class Mavenproject1{
    public static void main(String[] args)  {
        int a = 0, b = 0;
        Random r=new Random();
        for(int i=0;i<100;i++){
            b=r.nextInt(10);
            System.out.println("b= "+b);
            a=12345/b;
            System.out.println("a= "+a); // will not be printed if exception raised
        }
    }
}
```


Output



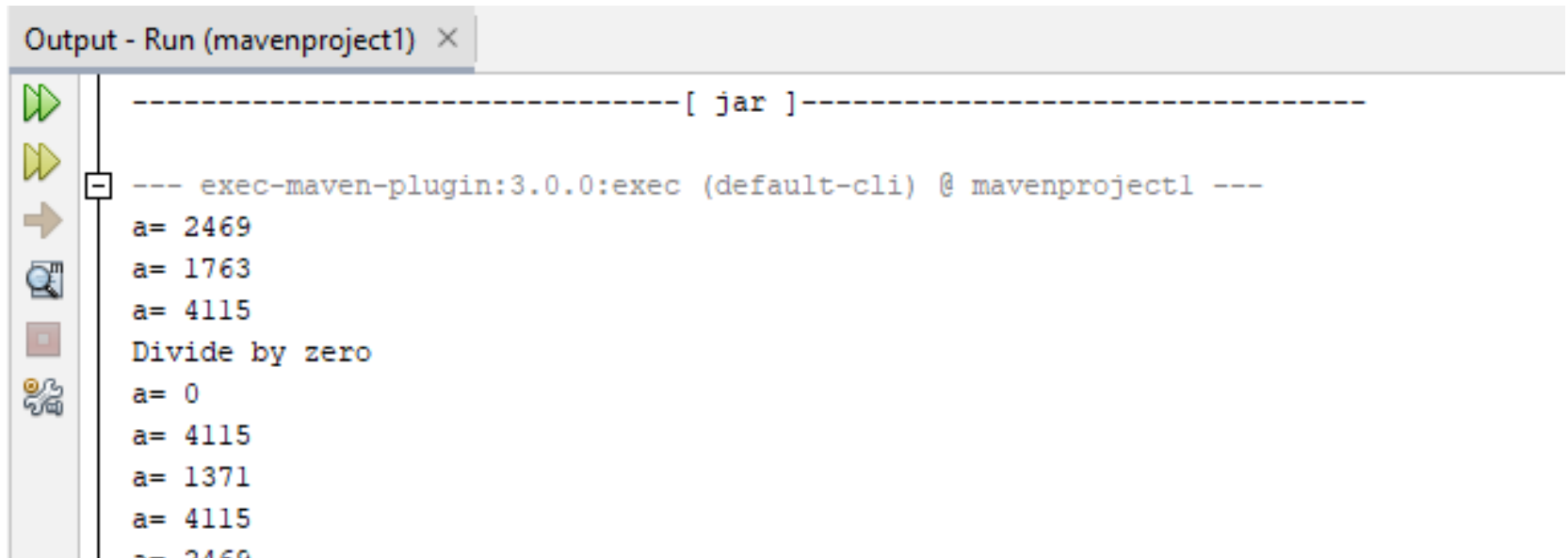
The screenshot shows an IDE's Output window titled "Output - Run (mavenproject1) X". The window contains a list of build output messages. On the left side of the window, there is a vertical toolbar with icons for running (green play button), stepping through (yellow play button), debugging (orange arrow), and other actions. The output messages are as follows:

- [-] Building mavenproject1 1.0-SNAPSHOT
 - [jar]-----
- [-] --- exec-maven-plugin:3.0.0:exec (default-cli) @ mavenproject1 ---
 - b= 7
 - a= 1763
 - b= 0
- [-] Exception in thread "main" java.lang.ArithmeticException: / by zero
 - [at com.mycompany.mavenproject1.Mavenproject1.main\(Mavenproject1.java:20\)](#)
- [-] Command execution failed.
- [-] org.apache.commons.exec.ExecuteException: Process exited with an error: 1 (Exit value: 1)

```
import java.util.Random;

public class Mavenproject1{
    public static void main(String[] args)  {
        int a = 0, b = 0;
        Random r=new Random();
        for(int i=0;i<100;i++){
            try{
                b=r.nextInt(10);
                a=12345/b;
            }
            catch(ArithmeticException e){
                System.out.println("Divide by zero");
                a=0;
            }
            System.out.println("a= "+a);        } }}
```

Output



```
-----[ jar ]-----  
--- exec-maven-plugin:3.0.0:exec (default-cli) @ mavenproject1 ---  
a= 2469  
a= 1763  
a= 4115  
Divide by zero  
a= 0  
a= 4115  
a= 1371  
a= 4115  
a= 2469
```

Exception handling in Java

```
public class JavaExceptionExample{  
    public static void main(String args[]){  
        try{  
            //code that may raise exception  
            int data=100/0;  
        }catch(ArithmeticException e)  
            {System.out.println(e.printStackTrace());}  
        //rest code of the program  
    }  
}
```

Output:
?

Different Exception handling

- Given some scenarios where unchecked exceptions may occur. They are as follows:
 1. A scenario where **ArithmeticException occurs**
 - If we divide any number by zero, there occurs an ArithmeticException.
 - `int a=50/0;//ArithmeticException`
 2. A scenario where **NullPointerException occurs**
 - If we have a null value in any variable, performing any operation on the variable throws a NullPointerException.
 - `String s=null;`
 - `System.out.println(s.length());//NullPointerException`

Different Exception handling...

3. A scenario where **NumberFormatException occurs**
 - ▣ The wrong formatting of any value may occur **NumberFormatException**. Suppose I have a string variable that has characters, converting this variable into digit will occur **NumberFormatException**.
 - ▣ `String s="abc";`
 - ▣ `int i=Integer.parseInt(s);//NumberFormatException`

4. A scenario where **ArrayIndexOutOfBoundsException occurs**
 - ▣ If you are inserting any value in the wrong index, it would result in **ArrayIndexOutOfBoundsException** as shown below:
 - ▣ `int a[]=new int[5];`
 - ▣ `a[10]=50; //ArrayIndexOutOfBoundsException`

Multiple Catch Clauses

- ❑ Two or more catches- each handles a different type of exception
- ❑ Each catch is inspected in order
- ❑ The first one whose type matches- execute it
- ❑ The remaining exceptions are bypassed
- ❑ Exception subclasses must come before any of their superclasses

□ Try this code:

```
int a[ ]={10};
```

```
i=1;
```

```
a[ i ] = i / (i-1);
```

Two catch statements in sequence:

ArrayIndexOutOfBoundsException

ArithmeticException

??

Nested try statements

- A try statement can be inside the block of another try

Throw and throws

- ❑ In previous examples, Java runtime exceptions
- ❑ Throw ThrowableInstance
- ❑ Mobile no program
- ❑ Method throws exception

Questions

- Difference between try-catch and if-else? With examples.
- Customized or user-defined exceptions.