COMP 250
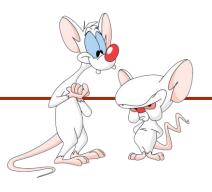# INTRODUCTION TO COMPUTER SCIENCE

Week 3-1: Errors, exceptions, and try catch blocks

Giulia Alberini, Fall 2020

WHAT ARE WE GOING TO DO IN THIS VIDEO?

- Errors and Exceptions

- Try/catch blocks

Assignment Project Exam Help

https://powcoder.com

Add WeChat powcoder

- **Stylistic Errors**
  The functionality of your code is not affected.
  Your code is hard to read.

- **Compile-time errors**
  Something is wrong with the syntax of the program.
  Your code does not compile!

- **Run-time errors**
  Something goes wrong when you run your program.
  Your code does not run!

- **Logic errors**
  Something isn't working the way you think.
  The program does not do the correct thing.

COMPILE TIME ERRORS

- These errors are detected by the compiler.

- They prevent your code from running since the program violates the syntactic rules of Java.

- Compile-time errors include missing a semicolon, adding an extra bracket, or using the incorrect types.

- Compile-time errors are the easiest to fix!

RUN-TIME ERRORS

- These errors happen at run time. They are detected by the JVM when it tries to execute the instructions in the program.

- Common run-time errors include index out of bounds exceptions, division by zero, or null pointer exceptions.

- They are more difficult to fix than compile-time errors, but the JVM output some useful text to help you with that.

Java displays a message with:

- the name of the exception,

- the line of the program where it occurred, and

- a "stack trace": the list of the method calls the application was in the middle of when an Exception was thrown

```
java.lang.StringIndexOutOfBoundsException: String index out of range: 3
    at java.lang.String.charAt(Unknown Source)
    at Test.method3(Test.java:19)
    at Test.method2(Test.java:13)
    at Test.method1(Test.java:9)
    at Test.main(Test.java:4)
    at sun.reflect.NativeMethodAccessorImpl.invoke0(Native Method)
```

THROWING EXCEPTIONS

An exception is an event which occurs during the execution of a program, that disrupts the normal flow of the program's instructions.

Exceptions happen when the computer is asked to do something that doesn't make sense.

```
int[] x = {1,2,3};
```
```
System.out.println(x[3]);
```

On the second line we try to access an element that does not exists. An `ArraysIndexOutOfBoundsException` will be thrown.

Assignment Project Exam Help

When you want to communicate to a user that an invalid

https://powcoder.com

input was passed, or that some error occurred during the

execution of the program, you can decide to deliberately

Add WeChat powcoder

stop the execution of your code by throwing an exception.

```
public static boolean isPrime (int n) {
    if (n <= 1) {
        return false;
    }
    ...
}
```

Do we want to return false? We can instead throw an exception!

```
public static boolean isPrime (int n) {
    if (n <= 1) {
        throw new IllegalArgumentException("Primality
is not defined for number smaller than 2");
    }
    ...
}
```

```
public static boolean isPrime (int n) {
    if (n <= 1) {
        throw new IllegalArgumentException("Primality
is not defined for number smaller than 2");
    }
    ...
}
```

Keywords

EXAMPLE

```java
public static boolean isPrime (int n) {
    if (n <= 1) {
        throw new IllegalArgumentException("Primality
is not defined for numbers smaller than 2");
    }
    ...
}
```

Keywords

Exception name

```
public static boolean isPrime (int n) {
    if (n <= 1) {
        throw new IllegalArgumentException("Primality
is not defined for a number smaller than 2");
    }
    ...
}
```

Message that will be displayed

THROWING EXCEPTIONS

You can `throw` any kind of Exception you want.

Example:

ArithmeticException, **ArrayIndexOutOfBoundsException**, ArrayStoreException, ClassCastException, **IllegalArgumentException**, IllegalMonitorStateException, IllegalStateException, IllegalThreadStateException, IndexOutOfBoundsException, NegativeArraySizeException, NullPointerException, NumberFormatException, …

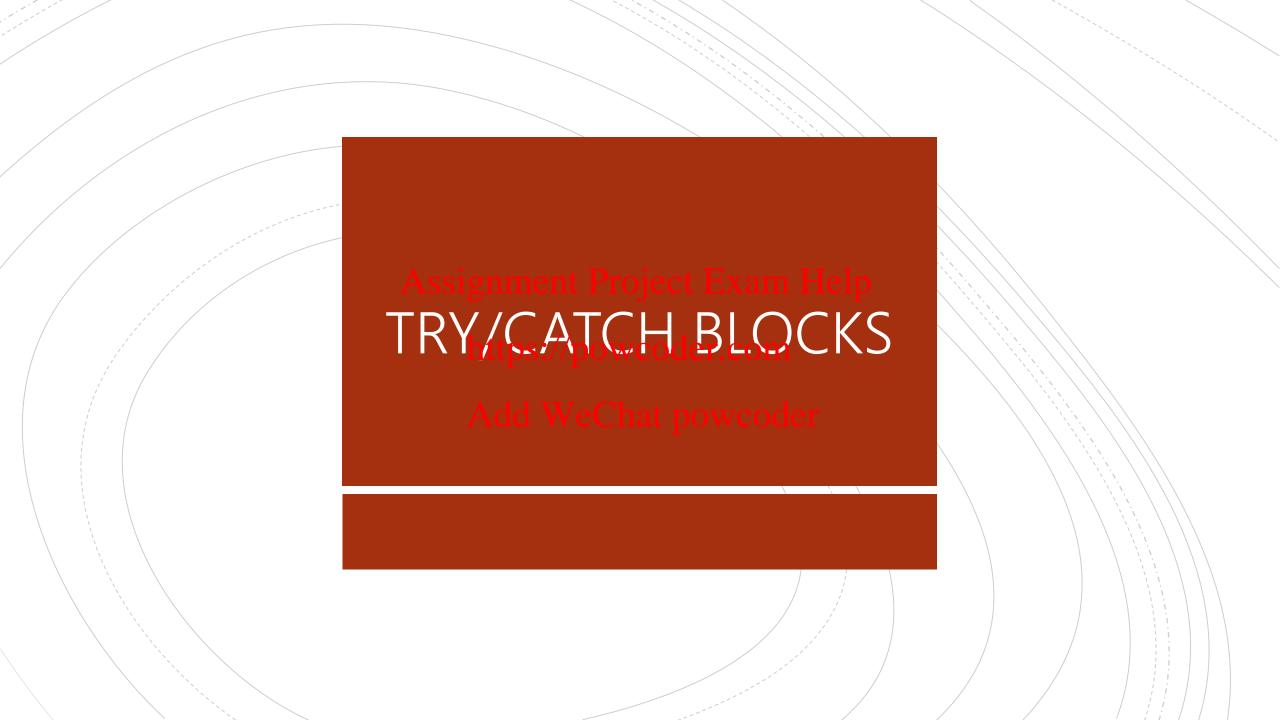In general, if you want to throw and exception you can add the following statement in your code:

```
throw new [name of exception] (message)
```

The type of message is `String` and the message is optional.

TRY/CATCH BLOCKS

What can we do to prevent our code from crashing at run-time?   Assignment Project Exam Help

```
public static void myMethod(int[] x) {
    if(x != null) {
        // something with x
    }
}
```

https://powcoder.com

Add WeChat powcoder

What type of exception are we trying to prevent?

Is there another way?

In order to prevent our code from crashing, we can *catch* Exceptions as follows:

```
try {
    // code that might be problematic
}
catch(Exception e) {
    // code that should be executed if there was a problem
}
// whatever comes after
```

e is the name I gave to my variable of type Exception

A try/catch block allows us to try some code, and if an exception is *raised*, we can then "catch" the exception. An exception that is caught will *not* cause the program to crash.

EXAMPLE 1

```
int[] x = {1,2,3};
try {
    System.out.println(x[3]);
}
catch(ArrayIndexOutOfBoundsException e) {
    System.out.println("Wrong index!");
}
System.out.println("Everything else");
```

Assignment Project Exam Help

https://powcoder.com

Add WeChat powcoder

What prints?

➤Wrong index!
  Everything else

EXAMPLE 2

```
int[] x = {1,2,3};
try {
    System.out.println(x[3]);
}
catch(NullPointerException e) {
    System.out.println("It is null!");
}
System.out.println("Everything else");
```

What prints?

➢ **Run-time error!**

`ArrayIndexOutOfBoundsException`

is thrown and it is not caught

by any catch block.

CATCHING MULTIPLE EXCEPTIONS

If we want to do something different depending on the type of the exception caught, we can use multiple catch blocks.

```
try {
    // code that might be problematic
}
catch(ArrayIndexOutOfBoundsException e) {
    // what to do in this case
}
catch(NullPointerException e) {
    // what to do in this other case
}
// whatever comes after
```

```
int[] x = {1,2,3};
try {
    System.out.println(x[1]/0);
}
catch(ArrayIndexOutOfBoundsException e) {
    System.out.println("Wrong index!");
}
catch(ArithmeticException e) {
    System.out.println("Bad math");
}
System.out.println("Everything else");
```

What prints?

➤Bad math

Everything else

If we would like to catch all possible exceptions with the same catch block we can do the following:

```
try {     https://powcoder.com
    // code that might be problematic
}         Add WeChat powcoder
catch(Exception e) {
    // what to do in case of issues
}
// whatever comes after
```

All different exceptions are considered to be of type `Exception`.

- If you would like to display all the information related to some caught exception (what you would normally see in red when the code crashes), you can use the following method:

```
e.printStackTrace();
```

- If you want to just print the name of the exception you can use a `print/println` statement:

```
System.out.println(e);
```

- The `finally` block **_always_** executes when the `try` block exists. This ensures that the `finally` block is executed even if one of the following happens:
    - an unexpected exception occurs in the `try` block
    - an exception occurs in the `catch` block
    - There's a `return/continue/break` statement in the `try/catch` block.
- You can have a `finally` even with just a `try` block (and no `catch`).
- The `finally` block is useful for more than just exception handling.
  Good practice: put cleanup code in a `finally` block even if no exception is anticipated.

CHECKED VS UNCHECKED EXCEPTIONS

CHECKED VS UNCHECKED EXCEPTIONS

In java there are two kinds of exceptions:

- **Checked**

  - Exception

  - IOException

  - FileNotFoundException

- **Unchecked**

  - NullPointerException

  - ArrayIndexOutOfBoundsException

  - ArithmeticException

- These exceptions are **_not_** checked at compile-time.

- Most (if not all) exceptions we have seen up to now are

  unchecked, they can cause your code to crash at run-time.

- You are not forced by the compiler to handle these exceptions. It is up to the programmer to decide if to catch the exceptions.

CHECKED EXCEPTIONS

- These exceptions are checked at compile-time!

- Usually, these kinds of Exceptions happen due to something that isn't the programmer's fault. For example, trying to read from a file whose name is misspelled. These are considered 'recoverable' errors, which is why you have to handle them - it might be possible to keep going and not just crash!

The programmer is forced to *handle* these exceptions. There are 2 ways to do that:

1. Use try/catch block to surround the code that might throw a checked exception

2. Specify in the method header that the method contains code that might throw an exception, and therefore the method itself (might) `throws` an exception.

EXAMPLE – READING FROM A FILE

You can read from a file in Java using the same `Scanner` class that we use to read from standard input!

You use an overloaded constructor inside the `Scanner` class.

`File` is an object meant to store information related to the File. You can create one by providing a path to a file.

Scanner

```
public Scanner(File source)
                  throws FileNotFoundException
```

Constructs a new Scanner that produces values scanned from the specified file.

Parameters:

    source - A file to be scanned

Throws:

    FileNotFoundException - if source is not found

EXAMPLE – READING FROM A FILE

Notice the `"throws"` keyword in the method header. This means that you need to *handle* a `FileNotFoundException` to avoid the compiler error.

Scanner

```
public Scanner(File source)
         throws FileNotFoundException
```

Constructs a new Scanner that produces values scanned from the specified file.

Parameters:

    source - A file to be scanned

Throws:

    FileNotFoundException - if source is not found

We would like to write the method below, but we get the following compile-time error:

Assignment Project Exam Help

```
unreported exception java.io.FileNotFoundException;
        must be caught or declared to be thrown
```

https://powcoder.com

Add WeChat powcoder

```
public static void myMethod(String filePath) {
    File f = new File(filePath);
    Scanner fileReader = new Scanner(f);
    int firstNumber = fileReader.nextInt();
}
```

Surround the code that might throw an exception with a try/catch block.

```java
public static void myMethod(int x) {
    try {
        File f = new File(filePath);
        Scanner fileReader = new Scanner(f);
        int firstNumber = fileReader.nextInt();
    }
    catch(FileNotFoundException e) {
        System.out.println("File not found");
    }

}
```

Specify in the method header that there's an exception using the `throws` keyword followed by the type of the exception.

```
public static void myMethod(String filePath) throws FileNotFoundException {
    File f = new File(filePath);
    Scanner fileReader = new Scanner(f);
    int firstNumber = fileReader.nextInt();
}
```

Whenever this method is called, the FileNotFoundException will have to be handled. To do so, you can use any of the two options we have just seen.

```java
public static void test() throws FileNotFoundException {
    // code from the previous slide
}

public static void test2() throws FileNotFoundException {
    :
    test();
    :
}

public static void test3() throws FileNotFoundException {
    :
    test2();
    :
}

public static void main(String[] args) throws FileNotFoundException {
    test3();
}
```

Assignment Project Exam Help

https://powcoder.com

Add WeChat powcoder

```java
public static void test() throws FileNotFoundException {
    // code from the previous slide
}


public static void test2() throws FileNotFoundException {
    :
    test();
    :
}



public static void test3() throws FileNotFoundException {
    :
    test2();
    :
}



public static void main(String[] args) {
    try{
        test3();
    } catch(FileNotFoundException e) {
        System.out.println("Caught here!");
    }
}
```

```
public static void test() throws FileNotFoundException {
    // code from the previous slide
}


public static void test2() throws FileNotFoundException {
    :
    test();
    :
}


public static void test3() {
    :
    try{
        test2();
    } catch(FileNotFoundException e) {
        System.out.println("Caught here!");
    }
    :
}


public static void main(String[] args) {
    test3();
}
```

Coming Soon

Assignment Project Exam Help

In the next video we will start to talk about objects and classes.
https://powcoder.com

Add WeChat powcoder