CYDEO

Exceptions

Exception

- An unwanted or unexpected event
- Occurs during the compile time or during the runtime
- There are two categories of exceptions: checked exception and unchecked exception
- To prevent exceptions from crashing our program, we must write code that detects and handles them

Unchecked Exceptions

- Exceptions that are not checked at compile time
- Occurs during the runtime
- Code will compile even if we do not handle them
- They have IS A relationship with RuntimeException (parent class

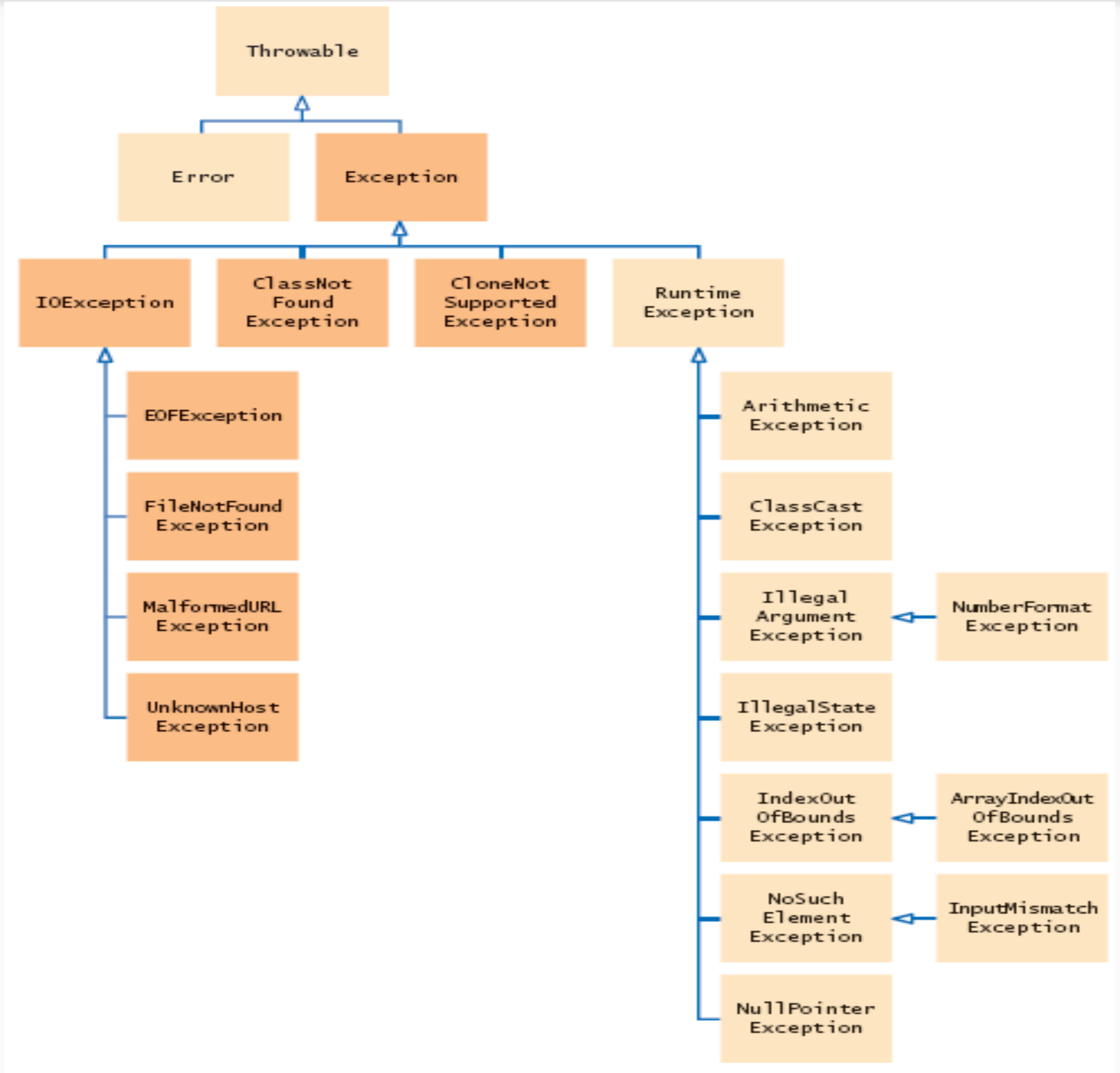
checked Exceptions

- Exceptions that are checked at compile time
- Occurs during the compile time
- Code will not compile even if we do not handle them
- They do not have IS A relationship with RuntimeException class

Errors

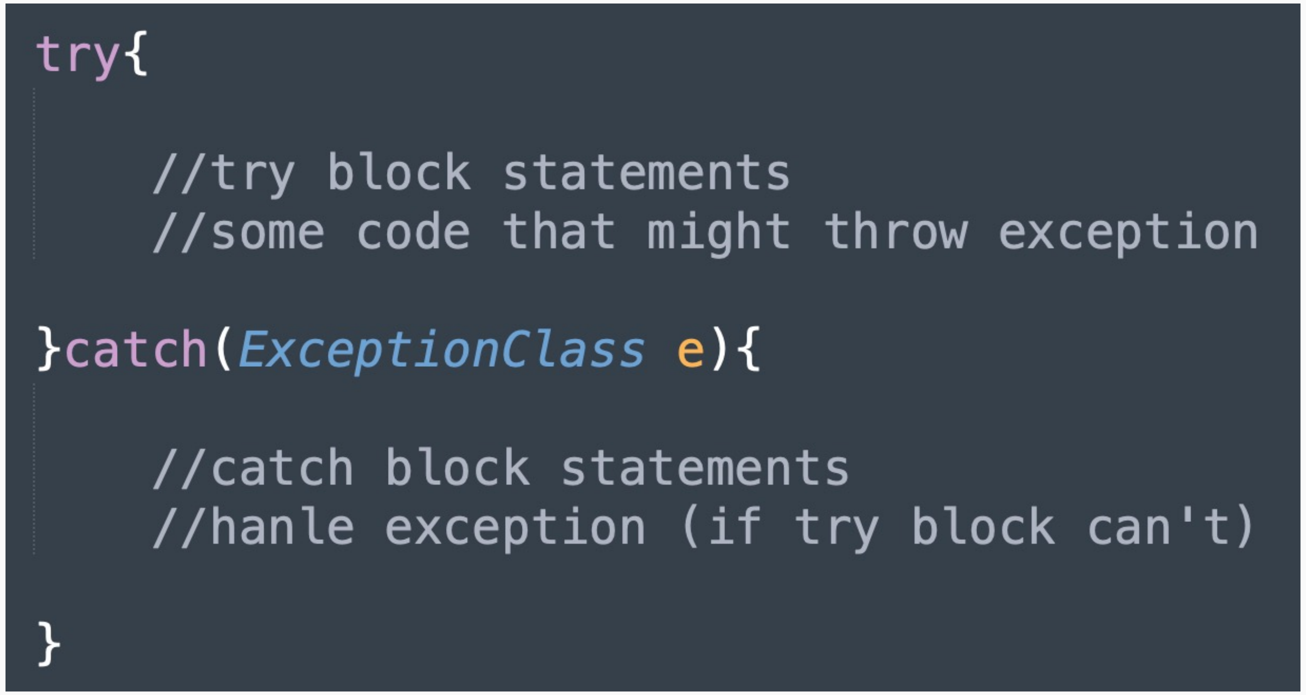
- Indicates that an illegal operation is being performed
- Occurs during the during the runtime only
- They can not be recovered, and not recommended to handle them

Exceptions/Errors Hierarchy



Exception Handling - try & catch

- To handle an exception (checked or unchecked), we can use try & catch blocks



```
try{

    //try block statements
    //some code that might throw exception

}catch(ExceptionClass e){

    //catch block statements
    //hanle exception (if try block can't)

}
```

CYDEO

Exception Object

- When runtime exception happens, java will catch it and assigns to a variable in catch block
- After it is successfully caught, we can use the variable and call some methods on the exception object
- Popular methods of exception objects are:
    - printStackTrace(): prints a stack trace (full details) of the exception
    - getMessage(): returns only brief description of the exception

Multiple catch Blocks

- If the code in the try block will be capable of throwing more than one type of exception
- To specify all the possible exceptions that could be thrown
- Parent exception class can not be placed before child exception class

```
try{

}catch(ArithmeticException e){

   //handle arithmetic exception

}catch(IndexOutOfBoundsException e){

   //handle index out of bounds exception

}catch(RuntimeException e){

   //handle Runtime exception

}
```

CYDEO

Finally block

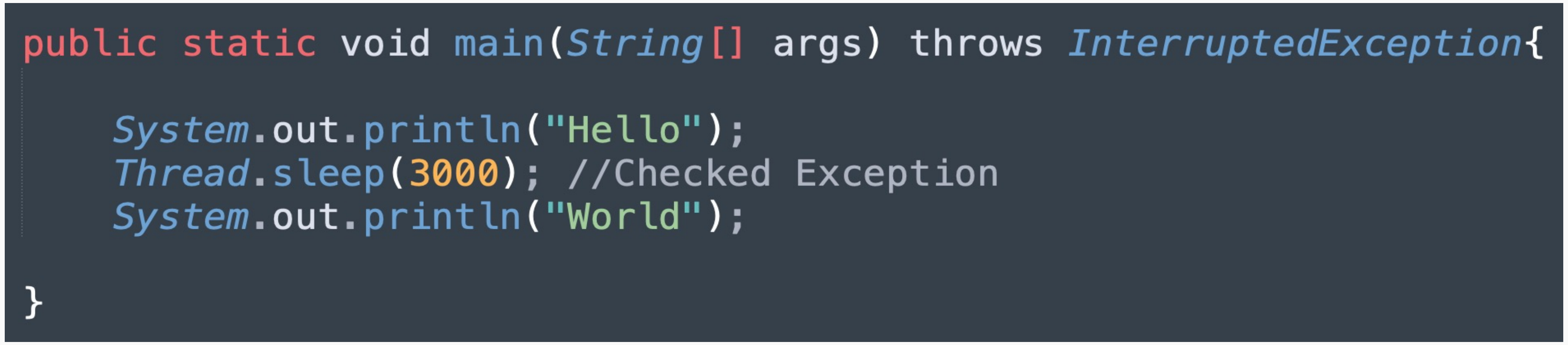
- An optional block that can be given after last catch block
- Always executed after try & catch blocks whether an exception occurs or not

```
try{
    //try block statements
}catch(ExceptionClass e){
    //catch block statements
}finally{
    // finally block statements
}
```

CYDEO

Throws Keyword

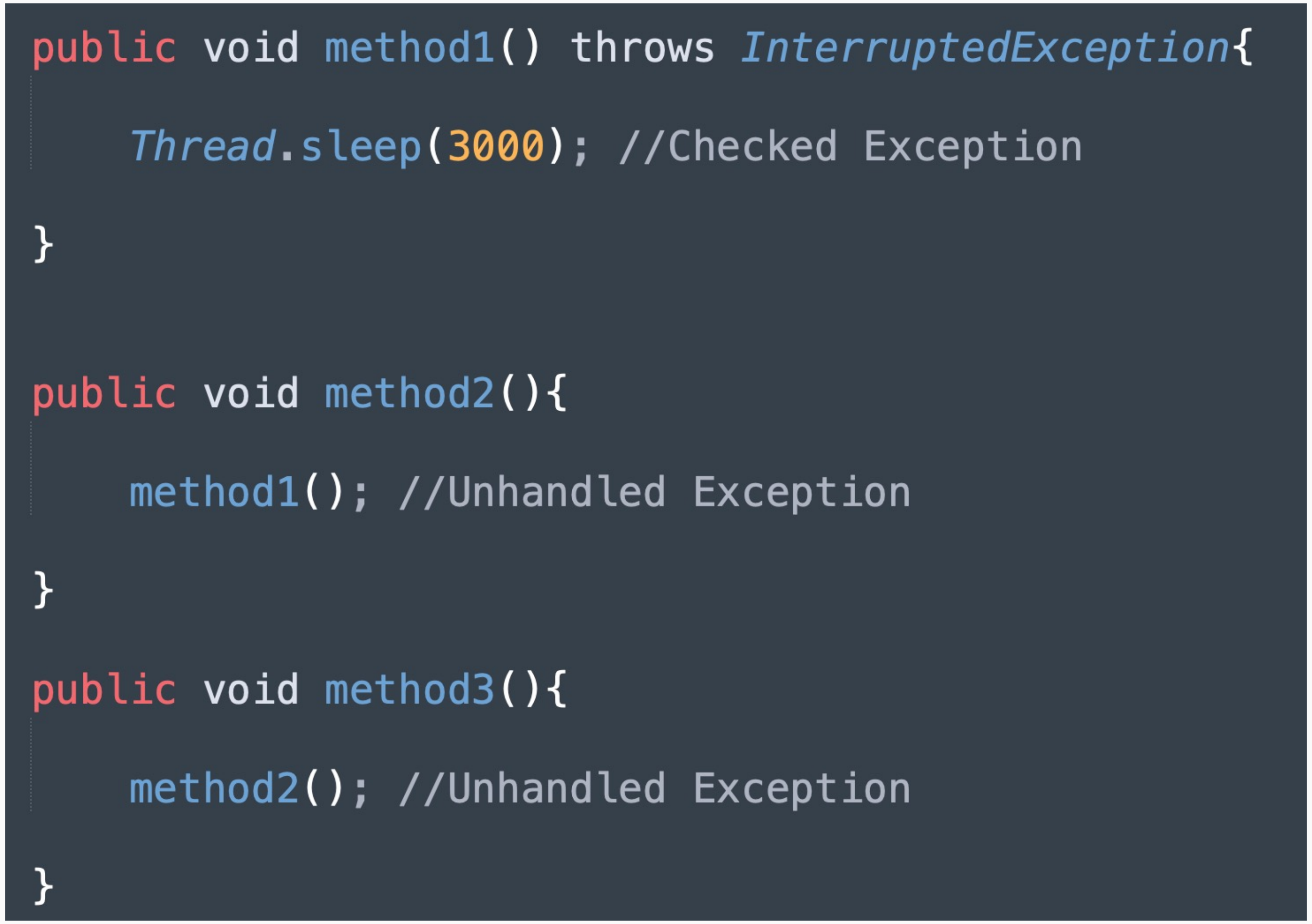
- Used within the method signature
- Informs the compiler that method throws one of the listed type exception
- Fastest way to get rid of the compilation error that's caused by a checked exception



```
public static void main(String[] args) throws InterruptedException{

    System.out.println("Hello");
    Thread.sleep(3000); //Checked Exception
    System.out.println("World");

}
```

Throws keyword - Rule

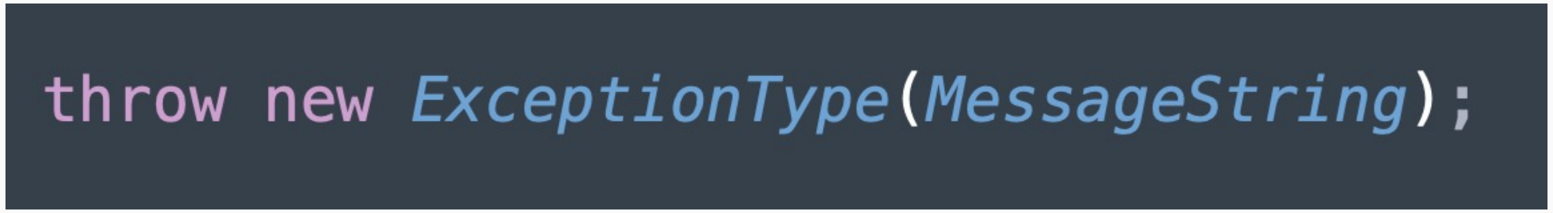
- Whoever calls the method that has throws keyword in its signature is responsible to handle it or declare it again



```java
public void method1() throws InterruptedException{

    Thread.sleep(3000); //Checked Exception

}


public void method2(){

    method1(); //Unhandled Exception

}

public void method3(){

    method2(); //Unhandled Exception

}
```

Throw keyword

- Used for manually throwing an exception



```
throw new ExceptionType(MessageString);
```