Checked and unchecked exceptions in java with examples

BY CHAITANYA SINGH | FILED UNDER: EXCEPTION HANDLING

There are two types of exceptions: checked exception and unchecked exception. In this guide, we will discuss them. The main **difference between checked and unchecked exception** is that the checked exceptions are checked at compile-time while unchecked exceptions are checked at runtime.

What are checked exceptions?

Checked exceptions are checked at compile-time. It means if a method is throwing a checked exception then it should handle the exception using try-catch block or it should declare the exception using throws keyword, otherwise the program will give a compilation error.

Lets understand this with the help of an **example**:

Checked Exception Example

In this example we are reading the file `myfile.txt` and displaying its content on the screen. In this program there are three places where a checked exception is thrown as mentioned in the comments below. FileInputStream which is used for specifying the file path and name, throws `FileNotFoundException`. The read() method which reads the file content throws `IOException` and the close() method which closes the file input stream also throws `IOException`.

```java
import java.io.*;
class Example {
   public static void main(String args[])
   {
        FileInputStream fis = null;
        /*This constructor FileInputStream(File filename)
         * throws FileNotFoundException which is a checked
         * exception
         */
        fis = new FileInputStream("B:/myfile.txt");
        int k;

        /* Method read() of FileInputStream class also throws
         * a checked exception: IOException
         */
        while(( k = fis.read() ) != -1)
        {
                System.out.print((char)k);
        }
        /*The method close() closes the file input stream
         * It throws IOException*/
        fis.close();
   }
}
```
Output:

```
Exception in thread "main" java.lang.Error: Unresolved compilation problems:
Unhandled exception type FileNotFoundException
Unhandled exception type IOException
Unhandled exception type IOException
```
Why this compilation error? As I mentioned in the beginning that checked exceptions gets checked during compile time. Since we didn't handled/declared the exceptions, our program gave the compilation error.
How to resolve the error? There are two ways to avoid this error. We will see both the ways one by one.

Method 1: Declare the exception using throws keyword.
As we know that all three occurrences of checked exceptions are inside main() method so one way to avoid the compilation error is: Declare the exception in the method using throws keyword. You may be thinking that our code is throwing FileNotFoundException and IOException both then why we are declaring the IOException alone. The reason is that IOException is a parent class of FileNotFoundException so it by default covers that. If you want you can declare them like this `public static void main(String args[]) throws IOException, FileNotFoundException`.

```
import java.io.*;
class Example {
   public static void main(String args[]) throws IOException
   {
      FileInputStream fis = null;
      fis = new FileInputStream("B:/myfile.txt");
      int k;

      while(( k = fis.read() ) != -1)
      {
           System.out.print((char)k);
      }
      fis.close();
   }
}
```
Output:
File content is displayed on the screen.

Method 2: Handle them using try-catch blocks.
The approach we have used above is not good at all. It is not the best exception handling practice. You should give meaningful message for each exception type so that it would be easy for someone to understand the error. The code should be like this:

```
import java.io.*;
class Example {
   public static void main(String args[])
   {
        FileInputStream fis = null;
        try{
            fis = new FileInputStream("B:/myfile.txt");
        }catch(FileNotFoundException fnfe){
           System.out.println("The specified file is not " +
                        "present at the given path");
         }
        int k;
        try{
            while(( k = fis.read() ) != -1)
            {
                 System.out.print((char)k);
            }
            fis.close();
        }catch(IOException ioe){
            System.out.println("I/O error occurred: "+ioe);
         }
   }
}
```

This code will run fine and will display the file content.

Here are the few other Checked Exceptions –

- SQLException
- IOException
- ClassNotFoundException
- InvocationTargetException

What are Unchecked exceptions?

Unchecked exceptions are not checked at compile time. It means if your program is throwing an unchecked exception and even if you didn't handle/declare that exception, the program won't give a compilation error. Most of the times these exception occurs due to the bad data provided by user during the user-program interaction. It is up to the programmer to judge the conditions in advance, that can cause such exceptions and handle them appropriately. All Unchecked exceptions are direct sub classes of **RuntimeException** class.

Lets understand this with an example:

Unchecked Exception Example

```
class Example {
   public static void main(String args[])
   {
        int num1=10;
        int num2=0;
        /*Since I'm dividing an integer with 0
         * it should throw ArithmeticException
         */
        int res=num1/num2;
        System.out.println(res);
   }
}
```

If you compile this code, it would compile successfully however when you will run it, it would throw `ArithmeticException`. That clearly shows that unchecked exceptions are not checked at compile-time, they occurs at runtime. Lets see another example.

```
class Example {
   public static void main(String args[])
   {
        int arr[] ={1,2,3,4,5};
        /* My array has only 5 elements but we are trying to
         * display the value of 8th element. It should throw
         * ArrayIndexOutOfBoundsException
         */
        System.out.println(arr[7]);
   }
}
```

This code would also compile successfully since `ArrayIndexOutOfBoundsException` is also an unchecked exception.

Note: It **doesn’t mean** that compiler is not checking these exceptions so we shouldn’t handle them. In fact we should handle them more carefully. For e.g. In the above example there should be a exception message to user that they are trying to display a value which doesn’t exist in array so that user would be able to correct the issue.

```
class Example {
   public static void main(String args[]) {
        try{
           int arr[] ={1,2,3,4,5};
           System.out.println(arr[7]);
        }
        catch(ArrayIndexOutOfBoundsException e){
           System.out.println("The specified index does not exist " +
                "in array. Please correct the error.");
        }
   }
}
```

Output:

```
The specified index does not exist in array. Please correct the error.
```

Here are the few unchecked exception classes:

- `NullPointerException`
- `ArrayIndexOutOfBoundsException`
- `ArithmeticException`
- `IllegalArgumentException`

- NumberFormatException