Exception Handling with Method Overriding in Java

Difficulty Level : Medium Last Updated : 09 Aug, 2021

An Exception is an unwanted or unexpected event, which occurs during the execution of a program i.e at run-time, that disrupts the normal flow of the program's instructions. Exception handling is used to handle runtime errors. It helps to maintain the normal flow of the program.
In any object-oriented programming language, Overriding is a feature that allows a subclass or child class to provide a specific implementation of a method that is already provided by one of its super-classes or parent classes. When a method in a subclass has the same name, same parameters or signature, and same return type(or sub-type) as a method in its super-class, then the method in the subclass is said to override the method in the super-class.

Exception Handling with Method Overriding
When Exception handling is involved with Method overriding, ambiguity occurs. The compiler gets confused as to which definition is to be followed.

Types of problems:

There are two types of problems associated with it which are as follows:

1. **Problem 1**: If The SuperClass doesn't declare an exception
2. **Problem 2**: If The SuperClass declares an exception

Let us discuss different cases under these problems and perceived their outputs.

Problem 1: If The SuperClass doesn't declare an exception

In this problem, two cases that will arise are as follows:

- **Case 1:** If SuperClass doesn't declare any exception and subclass declare checked exception
- **Case 2:** If SuperClass doesn't declare any exception and SubClass declare Unchecked exception

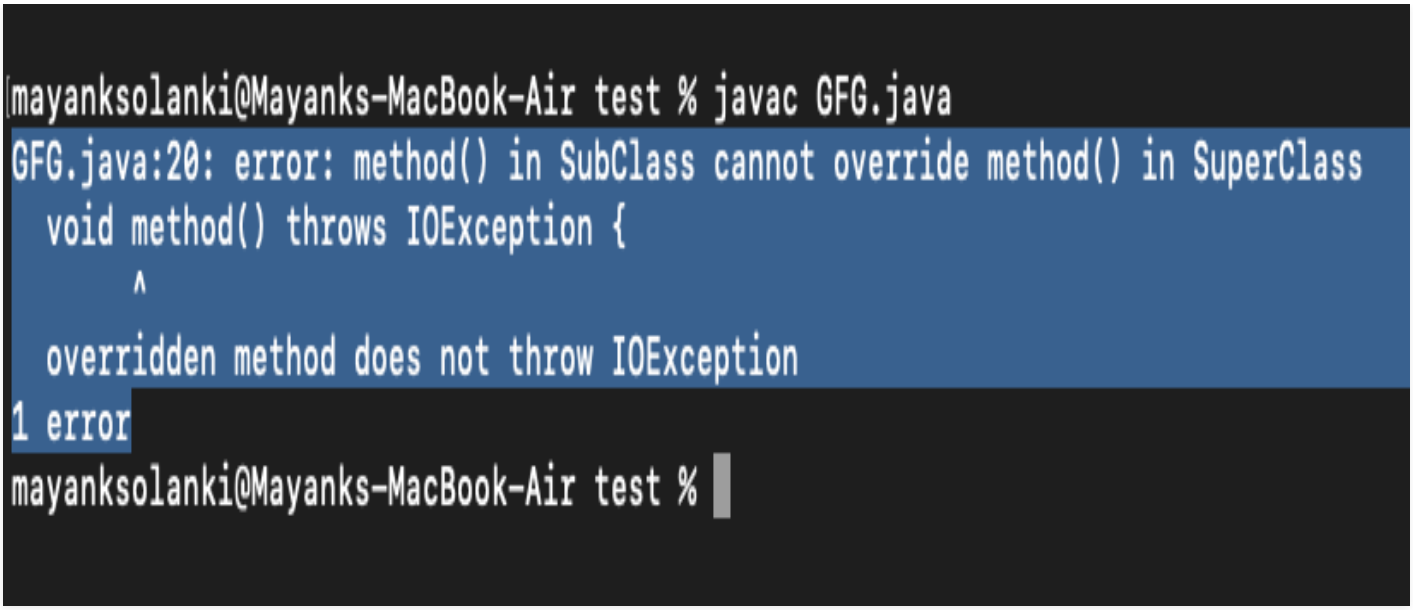
Let us discuss the above two cases and interpret them with help of examples as follows:

Case 1: If SuperClass doesn't declare any exception and subclass declare checked exception.

Example

```
// Java Program to Illustrate Exception Handling
// with Method Overriding
// Where SuperClass does not declare any exception and
// subclass declare checked exception

// Importing required classes
import java.io.*;

class SuperClass {

  // SuperClass doesn't declare any exception
  void method() {
    System.out.println("SuperClass");
  }
}

// SuperClass inherited by the SubClass
class SubClass extends SuperClass {

  // method() declaring Checked Exception IOException
  void method() throws IOException {

    // IOException is of type Checked Exception
    // so the compiler will give Error

    System.out.println("SubClass");
  }

  // Driver code
  public static void main(String args[]) {
    SuperClass s = new SubClass();
    s.method();
  }
}
```

Output:



Case 2: If SuperClass doesn't declare any exception and SubClass declare Unchecked exception

Example

```
// Java Program to Illustrate Exception Handling
// with Method Overriding
// Where SuperClass doesn't declare any exception and
// SubClass declare Unchecked exception

// Importing required classes
import java.io.*;

class SuperClass {

    // SuperClass doesn't declare any exception
    void method()
    {
        System.out.println("SuperClass");
    }
}

// SuperClass inherited by the SubClass
class SubClass extends SuperClass {

    // method() declaring Unchecked Exception ArithmeticException
    void method() throws ArithmeticException
    {

        // ArithmeticException is of type Unchecked Exception
        // so the compiler won't give any error
```

```
            System.out.println("SubClass");
        }

        // Driver code
        public static void main(String args[])
        {
            SuperClass s = new SubClass();
            s.method();
        }
    }
```

Output

```
SubClass
```

Now dwelling onto the next problem associated with that is if The SuperClass declares an exception. In this problem 3 cases will arise as follows:

- **Case 1:** If SuperClass declares an exception and SubClass declares exceptions other than the child exception of the SuperClass declared Exception.
- **Case 2:** If SuperClass declares an exception and SubClass declares a child exception of the SuperClass declared Exception.
- **Case 3:** If SuperClass declares an exception and SubClass declares without exception.

Now let us interpret these cases by implementing and interpreting with example.

Case 1: If SuperClass declares an exception and SubClass declares exceptions other than the child exception of the SuperClass declared Exception.

Example

```
// Java Program to Illustrate Exception Handling
// with Method Overriding
// Where SuperClass declares an exception and
// SubClass declares exceptions other than the child exception
// of the SuperClass declared Exception.

// Importing required classes
import java.io.*;
```

```
class SuperClass {

  // SuperClass declares an exception
  void method() throws RuntimeException {
    System.out.println("SuperClass");
  }
}

// SuperClass inherited by the SubClass
class SubClass extends SuperClass {

  // SubClass declaring an exception
  // which are not a child exception of RuntimeException
  void method() throws Exception {

    // Exception is not a child exception
    // of the RuntimeException
    // So the compiler will give an error

    System.out.println("SubClass");
  }

  // Driver code
  public static void main(String args[]) {
    SuperClass s = new SubClass();
    s.method();
  }
}
```

Output:

```
mayanksolanki@Mayanks-MacBook-Air test % javac GFG.java
GFG.java:18: error: method() in SubClass cannot override method() in SuperClass
  void method() throws Exception {
       ^
  overridden method does not throw Exception
1 error
mayanksolanki@Mayanks-MacBook-Air test %
```

Case 2: If SuperClass declares an exception and SubClass declares a child exception of the SuperClass declared Exception.

Example

```
// Java Program to Illustrate Exception Handling
// with Method Overriding
// Where SuperClass declares an exception and
// SubClass declares a child exception
// of the SuperClass declared Exception

// Importing required classes
import java.io.*;

class SuperClass {

    // SuperClass declares an exception
    void method() throws RuntimeException
    {
        System.out.println("SuperClass");
    }
}

// SuperClass inherited by the SubClass
class SubClass extends SuperClass {

    // SubClass declaring a child exception
    // of RuntimeException
    void method() throws ArithmeticException
    {

        // ArithmeticException is a child exception
        // of the RuntimeException
        // So the compiler won't give an error
        System.out.println("SubClass");
    }

    // Driver code
    public static void main(String args[])
    {
        SuperClass s = new SubClass();
        s.method();
    }
}
```

Output

```
SubClass
```

Case 3: If SuperClass declares an exception and SubClass declares without exception.

Example

```
// Java Program to Illustrate Exception Handling
// with Method Overriding
// Where SuperClass declares an exception and
// SubClass declares without exception

// Importing required classes
import java.io.*;

class SuperClass {

    // SuperClass declares an exception
    void method() throws IOException
    {
        System.out.println("SuperClass");
    }
}

// SuperClass inherited by the SubClass
class SubClass extends SuperClass {

    // SubClass declaring without exception
    void method()
    {
        System.out.println("SubClass");
    }

    // Driver code
    public static void main(String args[])
    {
        SuperClass s = new SubClass();
    try {
        s.method();
    } catch (IOException e) {
        e.printStackTrace();
    }
    }
}
```

Output

```
SubClass
```

Conclusions:

As perceived from above 3 examples in order to handle such exceptions, the followi ng conclusions derived are as follows:

- *If SuperClass does not declare an exception, then the SubClass can only declar e unchecked exceptions, but not the checked exceptions.*
- *If SuperClass declares an exception, then the SubClass can only declare the sa me or child exceptions of the exception declared by the SuperClass and any new Runtime Exceptions, just not any new checked exceptions at the same level or hi gher.*
- *If SuperClass declares an exception, then the SubClass can declare without exc eption.*