Catching Base and Derived Classes as Exceptions in C++ and Java

Difficulty Level : Easy • Last Updated : 02 Mar, 2023

An Exception is an unwanted error or hurdle that a program throws while compiling. There are various methods to handle an exception which is termed exceptional handling.

Let's discuss what is Exception Handling and how we catch base and derived classes as an exception in C++:

- If both base and derived classes are caught as exceptions, then the catch block of the derived class must appear before the base class.
- If we put the base class first then the derived class catch block will never be reached. For example, the following **C++** code prints ***"Caught Base Exception"***.

C++

```
// C++ Program to demonstrate a
// Catching Base Exception
#include <iostream>
using namespace std;

class Base {
};
class Derived : public Base {
};
int main()
{
    Derived d;
    // Some other functionalities
    try {
        // Monitored code
        throw d;
    }
    catch (Base b) {
        cout << "Caught Base Exception";
    }
    catch (Derived d) {
        // This 'catch' block is NEVER executed
        cout << "Caught Derived Exception";
    }
    getchar();
    return 0;
}
```

Output

```
Caught Base Exception
```

*The **output** of the above **C++** code:*

```
prog.cpp: In function ‘int main()’:
prog.cpp:20:5: warning: exception of type ‘Derived’ will be caught
     catch (Derived d) {
     ^
prog.cpp:17:5: warning:    by earlier handler for ‘Base’
     catch (Base b) {
```

In the above C++ code, if we change the order of catch statements then both catch statements become reachable.

AD

Following is the modified program and it prints "Caught Derived Exception"

C++

```
// C++ Program to demonstrate a catching of
// Derived exception and printing it successfully
#include <iostream>
using namespace std;

class Base {};
class Derived : public Base {};
int main()
```

```
{
    Derived d;
    // Some other functionalities
    try {
        // Monitored code
        throw d;
    }
    catch (Derived d) {
        cout << "Caught Derived Exception";
    }
    catch (Base b) {
        cout << "Caught Base Exception";
    }
    getchar(); // To read the next character
    return 0;
}
```

Output

```
Caught Derived Exception
```

Output:

```
Caught Derived Exception
```

In java, catching a base class exception before derived is not allowed by the compiler itself. In C++, the compiler might give a warning about it but compiles the code.

For example, *the following Java code fails in compilation with the error message* ***"exception Derived has already been caught"***

Java

```
// Java Program to demonstrate
// the error filename Main.java
class Base extends Exception {
}
class Derived extends Base {
}
public class Main {
    public static void main(String args[])
    {
        try {
            throw new Derived();
        }
        catch (Base b) {
        }
        catch (Derived d) {
        }
    }
}
```

Error:

```
prog.java:11: error: exception Derived has already been caught
    catch(Derived d) {}
```

In both C++ and Java, you can catch both base and derived classes as exceptions. This is useful when you want to catch multiple exceptions that may have a common base class.

In C++, you can catch base and derived classes as exceptions using the catch block. When you catch a base class, it will also catch any derived classes of that base class. Here's an example:

C++

```cpp
#include <iostream>
#include <exception>
using namespace std;

class BaseException : public exception {
public:
    virtual const char* what() const throw() {
        return "Base exception";
    }
};

class DerivedException : public BaseException {
public:
    virtual const char* what() const throw() {
        return "Derived exception";
    }
};

int main() {
    try {
        // code that might throw exceptions
        throw DerivedException();
    } catch (BaseException& e) {
        cout << "Caught exception: " << e.what() << endl;
    }
    return 0;
}
```

Output

```
Caught exception: Derived exception
```

In this example, a BaseException class is defined and a DerivedException class is derived from it. In the main() function, a DerivedException object is thrown. The catch block catches any BaseException object or derived object, and prints a message to the console indicating which exception was caught.

In Java, you can catch base and derived classes as exceptions using the catch block with multiple catch clauses. When you catch a base class, it will also catch any derived classes of that base class.

Here's an example:

Java

```java
class BaseException extends Exception {
    public BaseException() {
        super("Base exception");
    }
}

class DerivedException extends BaseException {
    public DerivedException() {
        super("Derived exception");
    }
}

public class ExceptionExample {
    public static void main(String[] args) {
        try {
            // code that might throw exceptions
            throw new DerivedException();
        } catch (DerivedException e) {
            System.out.println("Caught derived exception: " + e.getMessage());
        } catch (BaseException e) {
            System.out.println("Caught base exception: " + e.getMessage());
        }
    }
}
```

OUTPUT:

```
Caught derived exception: Derived exception
```

62

Related Articles