BridgeLabz

Employability Delivered

Java Core Concepts – Exceptions

Java Core Concepts

BridgeLabz
Employability Delivered

- Java Exceptions
- Java Reflections
- Java Annotations
- *Java Generics*
- *Java Properties*
- Java OpenCSV
- Java JSON using Gson

UC 1

Given a Message, ability to analyse and respond Happy or Sad Mood

- Continuation of Mood Analyser Problem in Junit Intro
- Create MoodAnalyser Object
- Call analyseMood function with message as parameter and return Happy or Sad Mood

Java Exceptions

BridgeLabz
Employability Delivered

Exceptions are needed to support Programs that encounters unexpected situations. This can be due to user enters bad inputs, network connection drops, database is not responding or disk is full, etc.

Java Exception Handling

BridgeLabz
Employability Delivered

Exceptions handling help the program

1. from stopping abruptly on one end and
2. at the other end report to users.

Java Exception Types

BridgeLabz

Employability Delivered

1. Checked Exceptions
   - A **checked exception** is an exception that occurs at the compile time, these are also called as **compile time exceptions**. These exceptions cannot simply be ignored at the **time of compilation**, the programmer should take care to handle these exceptions.
2. Unchecked Exceptions –
   - An unchecked exception is an exception that occurs at the time of execution. These are also called as **Runtime Exceptions.** An exception here is an object of a special class that implements the **java.lang.Throwable** interface.

UC 2

Handle Exception if User Provides Invalid Mood

- Like NULL

TC 2.1

Given Null Mood Should Return Happy

To make this Test Case pass Handle NULL Scenario using try catch and return Happy

Java Custom Exceptions

BridgeLabz
Employability Delivered

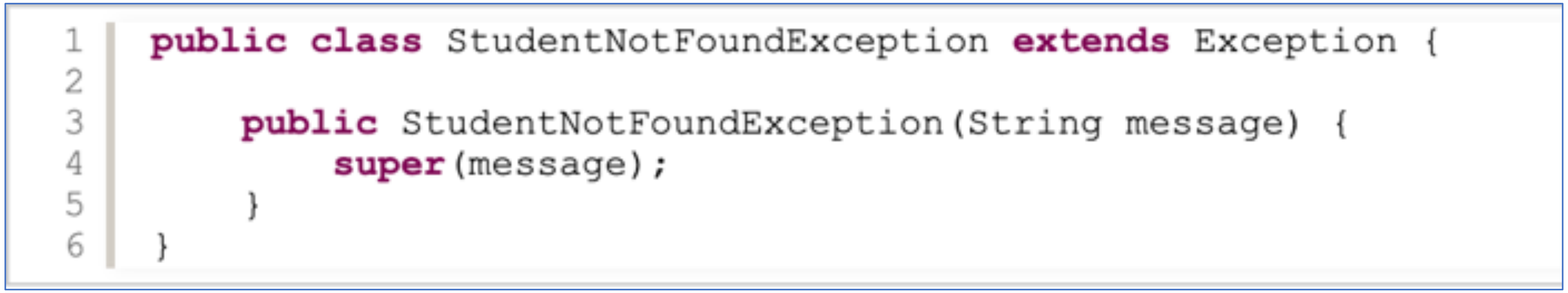
Custom Exceptions are needed to report to users the business exceptions which are at a level higher than the technical exceptions defined by Java.

Create Custom Exceptions

BridgeLabz
Employability Delivered

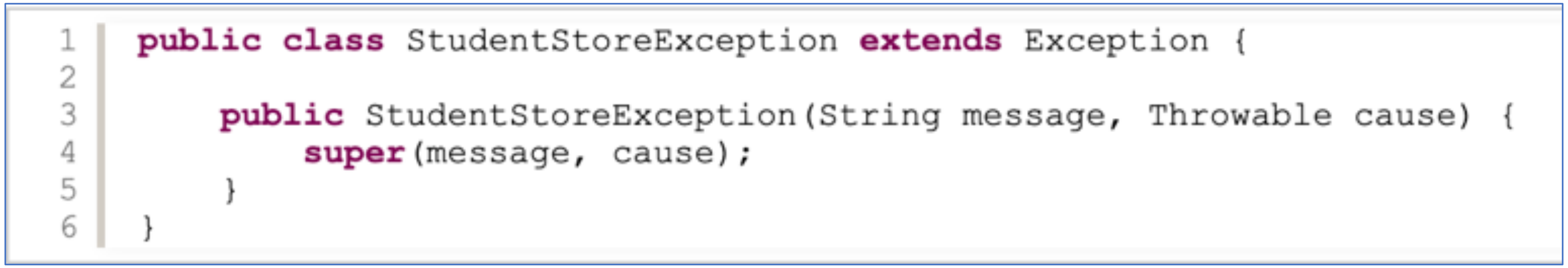
- Create a new class whose name should end with Exceptions like ClassNameException. This is a convention to differentiate an exception class from regular ones.
- Make the class extends one of the exceptions which are subtypes of the java.lang.Exception class. Generally, a custom exception class always extends directly from the Exception class.
- Create a constructor with a String parameter which is the detail message of the exception. In this constructor, simply call the super constructor and pass the message.
- Optionally Re-throw an exception by wrapping in a custom exception

Create Custom Exceptions



```java
public class StudentNotFoundException extends Exception {

    public StudentNotFoundException(String message) {
        super(message);
    }
}
```



```java
public class StudentStoreException extends Exception {

    public StudentStoreException(String message, Throwable cause) {
        super(message, cause);
    }
}
```

UC 3

Inform user if entered Invalid Mood

- Like in NULL or Empty Mood throw Custom Exception

TC 3.1

Given NULL Mood Should Throw InvalidMoodException

Handle NULL Scenario using try catch and throw InvalidMoodException

TC 3.2

Given Empty Mood Should Throw InvalidMoodException

Handle Empty Mood Scenario throw InvalidMoodException and inform user Empty or Null Mood

Guidance for Exception Handling

BridgeLabz
Employability Delivered

- Have a practice to Catch multiple exceptions then Generic Exceptions
- Remember the order of catch blocks does matter E..g FileNotFoundException is a child of IOException
- Do not recommend Catching one exception for all
- Avoid Grouping multiple exceptions in one catch unless the same exception is being reported.
- Typically all Reported Exceptions report Error Code and Message so the Client can Handle.
- E.g. HTTP Status or Error Codes

1xx Informational
2xx Success. ...

3xx Redirection. ...

4xx Client Error. ...
5xx Server Error. ...

BridgeLabz

Employability Delivered

Thank You