Exception Handling
Part 1

Prof. Siddharth Shah
Department of Computer Engineering
Dharmsinh Desai University

Outline

- Concept: Exception and Exception Handling
- Exception Generation
- Five Key Words
- General Form of Exception Handling Block
- Exception Class Hierarchy and Exception Classes
- **try** and **catch**

Concept: Exception and Exception Handling

- A Java exception is an object that describes an exceptional (that is, error) condition that has occurred in a piece of code.
- When an exceptional condition arises, an object representing that exception is created and thrown in the method that caused the error.
- That method may choose to handle the exception itself, or pass it on. Either way, at some point, the exception is caught and processed.

Exception Generation

- Exceptions can be generated by the Java run-time system, or they can be manually generated by your code.
- Exceptions thrown by Java relate to fundamental errors that violate the rules of the Java language or the constraints of the Java execution environment.
- Manually generated exceptions are typically used to report some error condition to the caller of a method

Five Key Words

- Java exception handling is managed via five keywords: **try, catch, throw, throws**, and **finally**.
- Program statements that you want to monitor for exceptions are contained within a try block. If an exception occurs within the **try** block, it is thrown.
- Your code can catch this exception (using **catch**) and handle it in some rational manner.

Five Key Words (Cont..)

- System-generated exceptions are automatically thrown by the Java runtime system. To manually throw an exception, use the keyword **throw**.
- Any exception that is thrown out of a method must be specified as such by a **throws** clause.
- Any code that absolutely must be executed after a try block completes is put in a **finally** block.

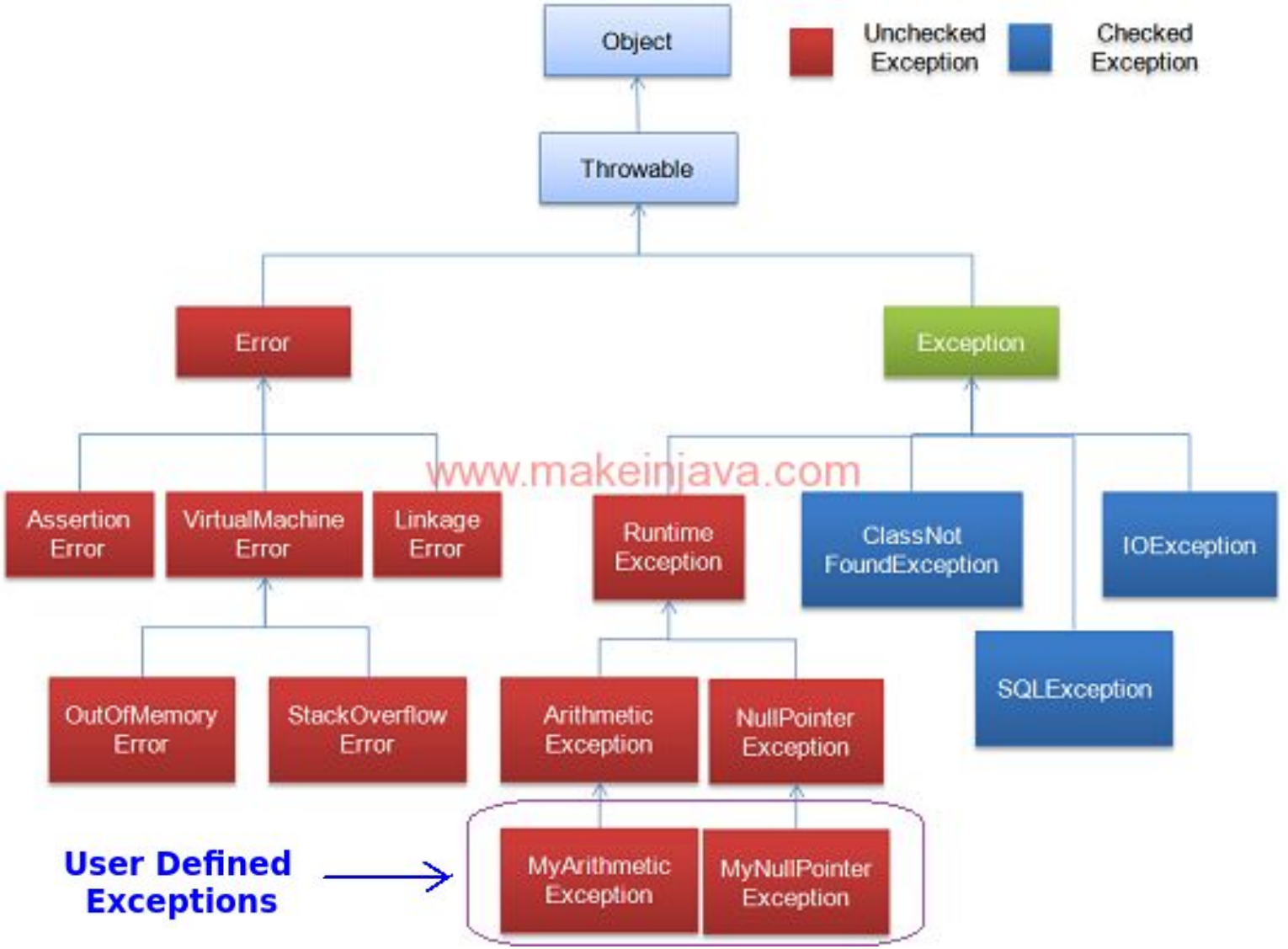
General Form of Exception Handling Block

```
try {
    // block of code to monitor for errors
}

catch (ExceptionType1 exOb) {
    // exception handler for ExceptionType1
}

catch (ExceptionType2 exOb) {
    // exception handler for ExceptionType2
}
// ...
finally {
    // block of code to be executed after try block ends
}
```

Exceptions Class Hierarchy



Exception Classes

- **Throwable:** All exception types are subclasses of the built-in class Throwable
- **Exception:** This class is used for exceptional conditions that user programs should catch. This is also the class that can be subclassed to create custom exception types.
- **RuntimeException:** It is an important subclass of Exception. Exceptions of this type are automatically defined for the programs and include things such as division by zero and invalid array indexing.

Exception Classes (Cont..)

- **Error:** It defines exceptions that are not expected to be caught under normal circumstances by the program.
- Exceptions of type Error are used by the Java run-time system to indicate errors having to do with the run-time environment, itself.
- Stack overflow is an example of such an error.
- We will not be dealing with exceptions of type Error, because these are typically created in response to catastrophic failures that cannot usually be handled by the program.

try and *catch*

- A **try** and its **catch** statement form a unit.
- The scope of the catch clause is restricted to those statements specified by the immediately preceding try statement.
- A catch statement cannot catch an exception thrown by another try statement (except in the case of nested try statements).
- The statements that are protected by try must be surrounded by curly braces. (That is, they must be within a block.) You cannot use try on a single statement.
- The goal of most well-constructed catch clauses should be to resolve the exceptional condition and then continue on as if the error had never happened.