Exception Handling

By,

Hitha Paulson

Assistant Professor, Dept. of Computer Science

LF College, Guruvayoor

Exception

- An exception is an abnormal condition that arises in a code sequence at run time
- A Java exception is an object that describes an exceptional condition that occurred in a piece of code
- When an exception arises, an exception object is created and thrown in the method that caused the error
- Generated exception should caught and processed by the code
- Exception can generated by
  - Java Runtime System (Errors caused by violation of language rules or by constraints of execution environment)
  - Manually generated by user code (Used to report some error condition to the caller of a method)

Exception Handling

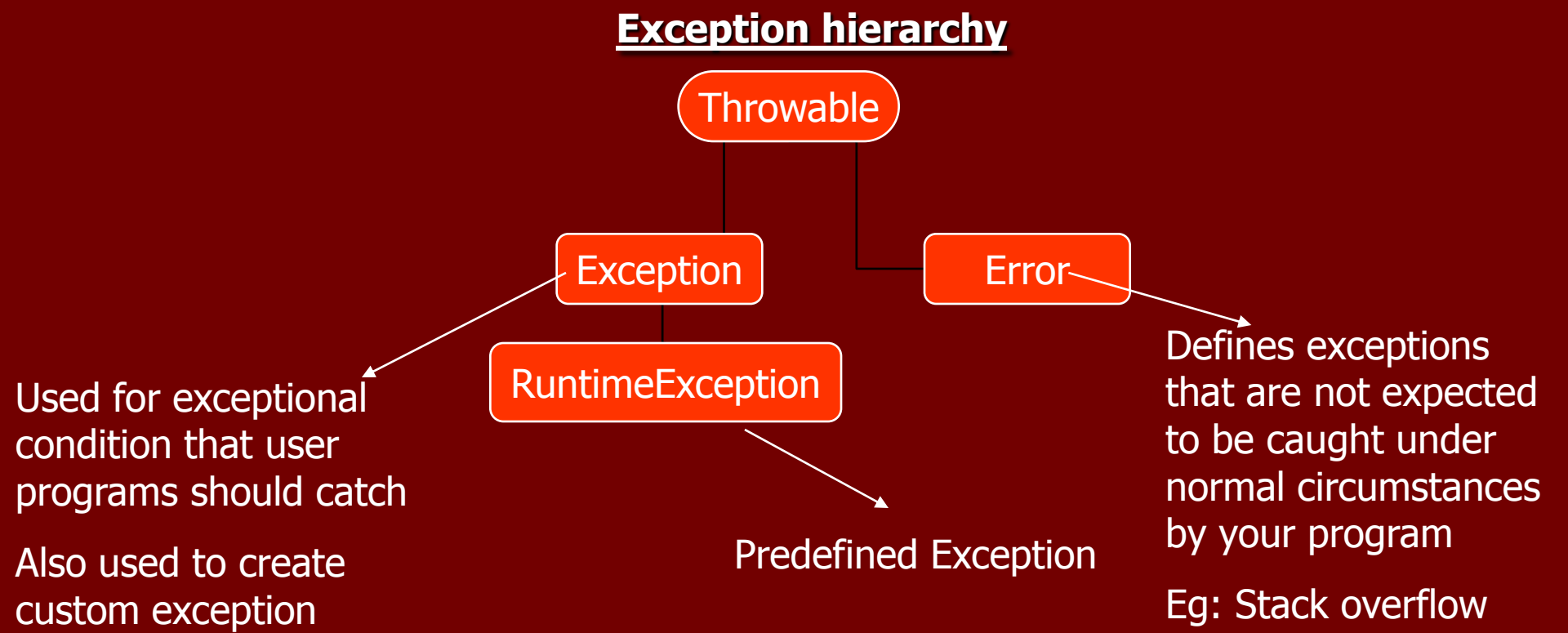
- Step 1: Program statements that can generate error will enclose within a **try** block
- Step 2: When exception occurs within try block, it will **thrown**
  - System generated exceptions are automatically thrown by Java Runtime
  - Manual exceptions should explicitly thrown by **throw** statement
  - Any exception thrown out of a method must be specified by using **throws** clause
- Step 3: Generated exception should catch and handle by user code
- Step 4: Any code that must execute after try block is put in **finally** block

General Form

```
try
{
    //block of code to monitor for errors
}
catch(ExceptionType1  exObj)
{
    //exception handler for Exception type1
}
catch(ExceptionType2  exObj)
{
    //exception handler for Exception type2
}
// ……..
finally
{
    //block of code to be executed after try block ends
}
```

Exception Types

- All exception types are subclasses of the built-in class **Throwable**



Uncaught Exceptions

- When an exception is not handled by the user, Java run-time system constructs a new exception objects and throws
- Unhandled exception causes termination of program execution
- Uncaught exception will handle by default handler
- Default handler displays a string describing the exception, prints a stack trace from which the point the exception occurred

Using try and catch

- Exception handling helps to
  - Fix the error
  - Prevents the program from automatically terminating
  - ie) to resolve the exceptional condition and to continue as if the error had never happened
- When an exception occurs inside the **try** block, control will transfer to **catch** block
- After executing **catch** block, execution continues with rest of the statements after **try..catch** block
- **try..catch** forms a unit and all try should have atleast one catch block
- **Catch** statement cannot catch an exception thrown by another block of **try..catch** statement

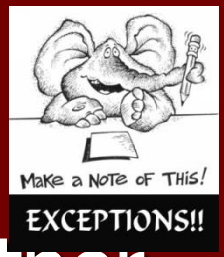
Exception Object

- An exception object can directly print by using println() method.
- A toString() method will automatically invoke to get exception description from exception object

Multiple catch Clauses

- In order to handle multiple exception generated from a **try** block, a try block can preceed more than one **catch** block
- Multiple **catch** block will contain different Exception classes
- Exception sub-classes should come before exception superclass
- When an exception is thrown,
  - Each **catch** block will inspect to find a matching block for the generated exception
  - Statements in the matching **catch** block will execute
  - Bypass rest of the **catch** blocks
  - Execute statements in **finally** block and goes to the immediate statement in the sequence

Nested try statements



- A **try** statement inside the block of anotner **try** statement
- If an inner **try** block does not have a matching **catch** block for a particular exception
  - The next **try** statement's catch block will inspected for a match
  - Above process continues until a match is found or exited from nested **try** block
  - If no matching **catch** found, default exception handler will take over the duty

Throw statement

- Manually generated exceptions are thrown by using **throw** statement
- General Format
  - throw ThrowableInstance;
  - ThrowableInstance is an instance of Throwable class or subclass of Throwable
- Primitive types and non-Throwable classes cannot use as exception
- Throwable object are created from parameter in catch block or by using **new** operator
- Flow of execution stops after throw statement, nearest try block is inspected to see matching catch block

Throws statement

- A method capable of causing an exception can defined using throws clause in the method's declaration
- It helps the caller of the method to guard themself against the exception
- A throws clause lists the types of exceptions that a method might throw
- Error, RuntimeException or any of their subclasses are not used with throws
- All other exceptions that a method can throw must be declared in the throws clause

General Format

Type method_name(parameter-list) throws exception-list

{

//body of method

}

- exception-list is a comma separated list of the exceptions that a method can throw

finally

- Finally creates a block of code that will be executed after a try/catch block has completed and before the code following the try/catch block
- Finally block will execute whether or not an exception is thrown
- Finally block is optional and each try statement requires atleast one catch or finally block

Self Study

- Built-in Exceptions
- User defined exceptions