Exception Handling

MODULE NAME: OBJECT ORIENTED PROGRAMMING WITH JAVA

MODULE CODE: CST 05206

INSTRUCTORS: JAMILA L.

Errors

Error is irrecoverable. Some example of errors are OutOfMemoryError, VirtualMachineError, AssertionError etc.

There are three categories of errors: syntax errors, runtime errors, and logical/semantic errors.

- Syntax errors arise because the rules of the language have not been followed.
- Syntax errors are easy to detect as compiler displays the error message.
- E.g., forgetting semicolon to terminate statement, using uninitialized variable, etc.

Errors cont…

Runtime errors:

Occur while the program is running if the environment detects an operation that is impossible to carry out.

- Runtime error causes the program to terminate immediately without successfully performing its job and display an error message.
- E.g., division by zero, assigning a variable to the wrong type of variable, etc.

Errors cont…

Logic errors:

Occur when a program does not perform the way it was intended to.

- Logic errors are difficult to detect because compiler does not display error messages.
- E.g. %d instead of %f for float variable using printf() function in C.

Introduction

- An **exception** is an indication of a problem that occurs during a program's execution.
- Exception is an event that disrupts the normal flow of the program.
- It is an object which is thrown at runtime.
- Exception handling in Java provides the powerful mechanism to handle the runtime errors so that normal flow of the application can be maintained.
- Exception handling enables you to create applications that can resolve (or handle) exceptions.

Introduction cont...

- Handling an exception allows a program to continue executing as if no problem had been encountered.
- Programs are fault-tolerant if they are able to deal with problems that may arise and continue executing.
- The core advantage of exception handling is to maintain the normal flow of the application.

What is happening when there is noExeption handler?

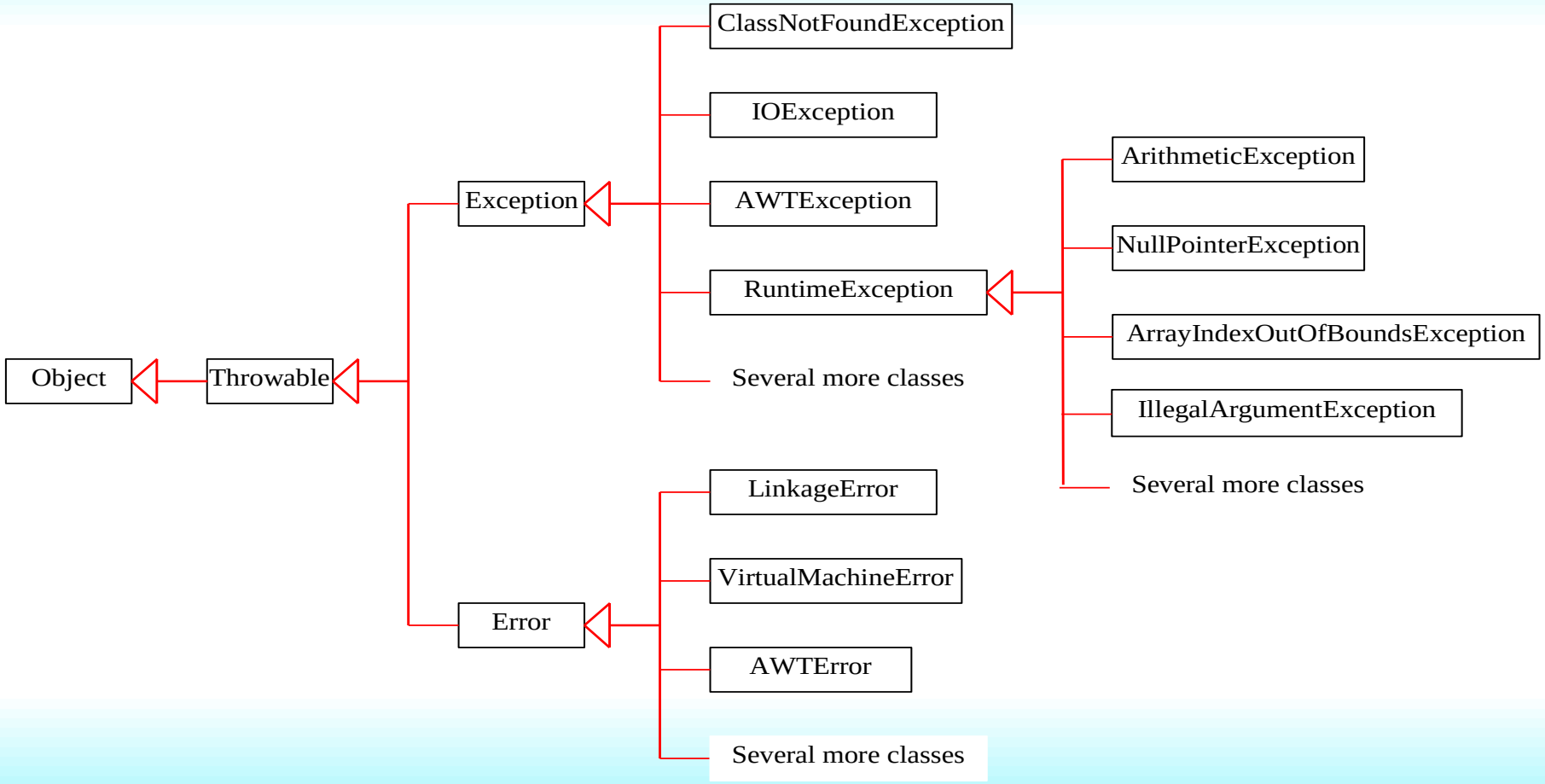
```
public class Noexception {

    public static void main(String args[]){
        int quotient;
           quotient=100/0;

        System.out.println("The quotient is: " +
quotient);

}
}
```

Hierarchy of Java Exception classes



Hierarchy cont…

- Throwable class is a super class of all exceptions and errors.
- Exceptions has a special subclass, the RuntimeException
- A user defined exception should be a subclass of the exception class

Types of exceptions

- There are two types of exceptions:
  1. Checked exception
  2. Unchecked exception
- The classes that extend Throwable class except RuntimeException and Error are known as checked exceptions.

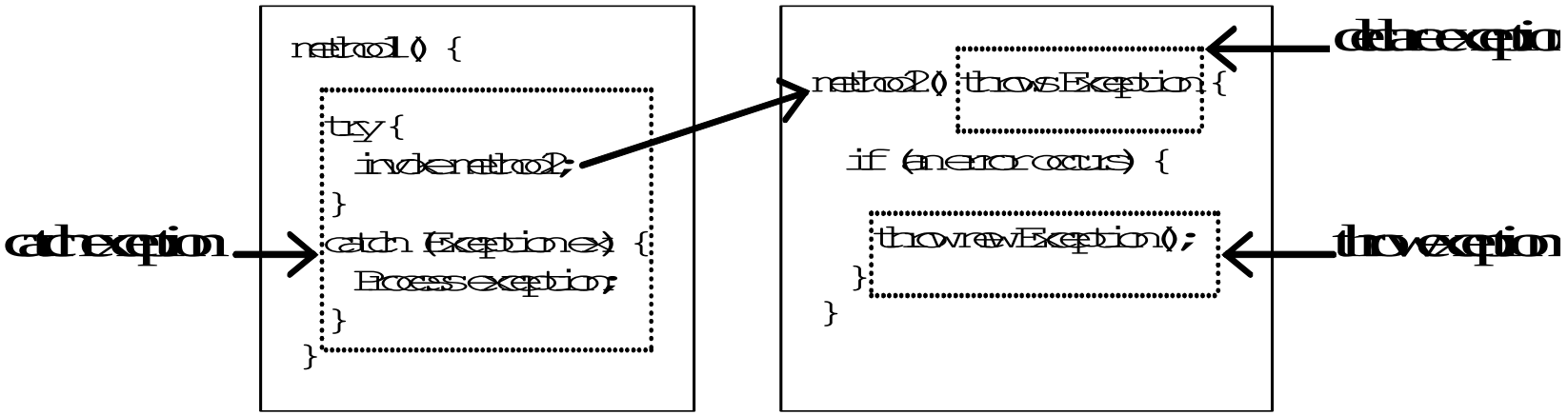
Types of exceptions cont...

- Examples of checked exceptions are IOException, SQLException, etc.
- Checked exceptions are checked at compile-time.
- All checked exceptions must be handled in the program.

Types of exceptions cont...

- The classes that extend RuntimeException and Error are known as unchecked exceptions
- Examples of unchecked exceptions are ArithmeticException, ArrayIndexOutOfBoundsException, etc.
- Unchecked exceptions are not checked at compile-time rather they are checked at runtime.

Declaring, throwing, and catching exceptions



Declaring exceptions

- Every method must state the types of checked exceptions it might throw.
- This is known as declaring exceptions.
- Examples:
  - `public void myMethod() throws IOException{}`
  - `public void myMethod() throws IOException, OtherException{}`

Throwing exceptions

- When the program detects an error, the program can create an instance of an appropriate exception type and throw it.
- This is known as throwing an exception.
- The throw clause can be used in any part of code where you feel a specific exception needs to be thrown to the calling method.
- E.g., throw new TheException(); or
  - TheException ex = new TheException();

  throw ex;

Throwing exceptions cont…

```
/** Set a new radius */
  public void setRadius(double newRadius)
      throws IllegalArgumentException {
    if (newRadius >= 0){
      radius =  newRadius;
    }else{
      throw new IllegalArgumentException(
        "Radius cannot be negative");
  }
}
```

Catching exceptions

- This is done by using try…catch block.
- A try block encloses the code that might throw an exception and should not execute if an exception occurs.
- A catch block catches (i.e., receives) and handles an exception.
- At least one catch or a finally block must immediately follow the try block.
- Therefore, a try block cannot stand alone.
- Each catch block specifies in parentheses an exception parameter that identifies the exception type the handler can process.

Catching exceptions cont…

- You can catch multiple exceptions that might arise in try block.
- When an exception occurs in a try block, the catch block that executes is the one whose type matches the type of the exception that occurred.

Syntax for try…catch block

```
try {
  //Statements that may throw exceptions
}
catch (Exception1 exVar1) {
  //handler for exception1;
}
catch (Exception2 exVar2) {
  //handler for exception2;
}
...
catch (ExceptionN exVarN) {
  //handler for exceptionN;
}
```

With Exception handler

```
public class Division{
    static  int quotient;
     public static void main(String args[]){
          try{
              //code that may raise exception
              quotient=50/0;
          }catch(ArithmeticException e){
               System.out.println("Error, no divisor 0");
          }
          //rest code of the program
          System.out.println("The quotient is: " +
quotient);
          }
}
```

Example

```
/*The program handles input mismatch by creating
robust and fault-tolerant program*/
import java.util.*;
public class ExceptionHandling {
    static int a,b;//static variable
    static int add(){
        return a+b;
    }
}
```

Example cont…

```
class ArithmeticTest{
    public static void main(String[] args){
        Scanner input=new Scanner(System.in);
        //declare boolean variable would exit when
true and continue when false
        boolean go=true;
        while(go){
```

Example cont…

```
try{
        System.out.println("Enter the first number");
        ExceptionHandling.a=input.nextInt();
        System.out.println("Enter the second number");
        ExceptionHandling.b=input.nextInt();
        System.out.println("The sum of two numbers is
"+ExceptionHandling.add());
        go=false;
        }//end try block
```

Example cont…

```
 catch(InputMismatchException wrong){
                System.out.println("Wrong input, try
again!");
                input.next();//discard input
            }//end catch block
            } //end while loop
        }//end main
 }//end class ArithmeticTest
```

Thank you!

for listen