GIA DINH
UNIVERSITY

Chapter 5

Exception - API - File handling

Objectives

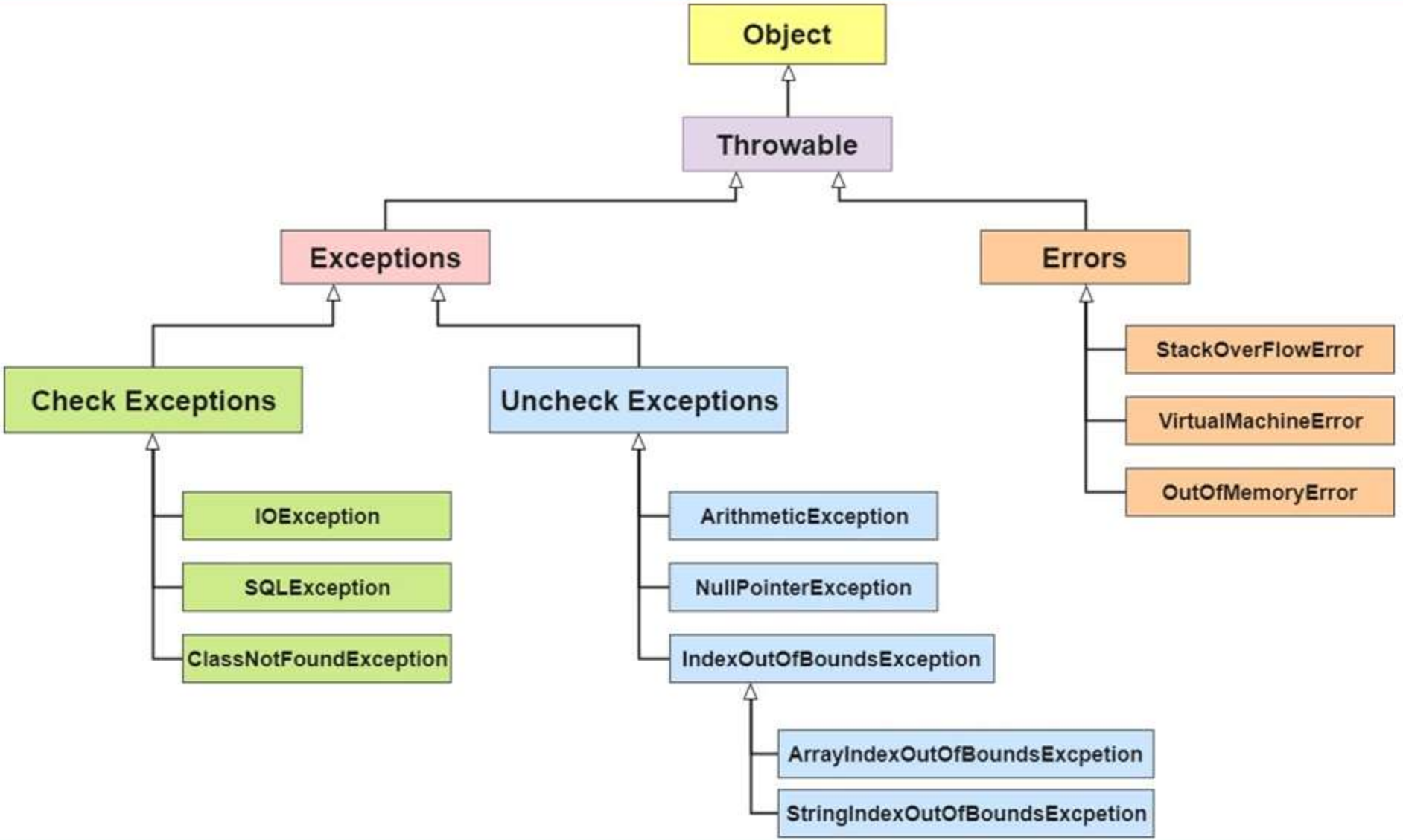
- *Exception*
- *API*
- *File handling*

Exception

Exceptions

- Different errors:
  - coding errors
  - wrong input
  - others
- Java will throw an error → exception

Exceptions hierarchy in Java



try … catch … finally

- Syntax

```
try {
  //  Block of code to try
} catch (Exception_class_Name_1  ex) {
  //  Block of code to handle errors 1
} catch (Exception_class_Name_2  ex) {
  //  Block of code to handle errors 2
} catch (Exception_class_Name_n  ex) {
  //  Block of code to handle errors n
} finally { //  Block of code
}
```

Example

```
public static void main(String[] args) {
        try {
                int [] a = new int[5];
                a[10] = 6;
                System.out.println("a[10] = " + a[10]);

                int zero = 0;
                int average = 10 / zero;
                System.out.println("Average = " + average);

                String obj = null;
                System.out.println(obj.length());
        } catch (NullPointerException ex) {
                System.out.println(ex);
        } catch (ArithmeticException ex) {
                System.out.println(ex);
        } catch (ArrayIndexOutOfBoundsException ex) {
                System.out.println(ex);
        } catch (Exception ex){
                System.out.println(ex);
        } finally {
                System.out.println("Finished!");
        }
}
```

throw

- throw: to create a custom error
- Syntax: throw new Exception_type(" ");

  Exception_type:

  ArithmeticException

  ArrayIndexOutOfBoundsException

  FileNotFoundException


```
public class Test{
  static void checkAge(int age) {
     if (age < 16) {
        throw new ArithmeticException("Access denied-You must be at least 16 years old.");
     } else
              System.out.println("Access granted - You are old enough!");
   }

   public static void main(String[] args) {
      checkAge(12);
   }
}
```

API

Java API

- API: Application Programming Interface
- Java API:
  - a library of prewritten classes
  - contains components for managing input, database programming…
- Syntax
  - import *package.name.Class*; // Import a single class
  - import *package.name.**; // Import the whole package

User input: Scanner

- Ex

```
import java.util.Scanner;
class Test {
  public static void main(String[] args) {
    Scanner input = new Scanner(System.in);
    System.out.println("Enter name, age and
height:");

    String name = input.nextLine(); // String input


    int age = input.nextInt(); // Numerical input
    float height = input.nextFloat();

    // Output input by user
    System.out.println("Name: " + name);
    System.out.println("Age: " + age);
    System.out.println("Height: " + height);
  }
}
```

Date and Time

- Java does not have a built-in Date class
- import java.time.*; // package to work with the date and time API

```
import java.time.LocalDate;
class Test {
   public static void main(String[] args) {
            LocalDate localDate = LocalDate.now(); // new LocalDate() → error
            System.out.println(localDate);
   }
}
```

Class	Description
LocalDate	yyyy-MM-dd
LocalTime	HH-mm-ss-ns
LocalDateTime	yyyy-MM-dd-HH-mm-ss-ns
DateTimeFormatter	Formatter for displaying

Formatting Date and Time

Value	Example
yyyy-MM-dd	"2039-09-19"
dd/MM/yyyy	"19/09/2039"
dd-MMM-yyyy	"19-Sep-2039"
E, MMM dd yyyy	“Mon, Sep 19 2039"

- Ex:

```
import java.time.*;
import java.time.format.*;
class Test {
   public static void main(String[] args) {
            LocalDateTime localDateTime = LocalDateTime.now();
            System.out.println("Before formatting: " + localDateTime);

      DateTimeFormatter formatter = DateTimeFormatter.ofPattern("HH:mm dd/MM/yyyy");
      System.out.println("After formatting: " + localDateTime.format(formatter));
   }
}
```

Wrapper

- Wrapper
  - classes
  - primitive data types (byte, short, float, boolean, …) as objects
  - to use certain methods

Primitive Data Type	Wrapper Class
byte	Byte
short	Short
int	Integer
long	Long
float	Float
double	Double
boolean	Boolean
char	Character

Wrapper

- Ex:

```
public class Test {
        public static void main(String[] args) {
                Float f = 1.123f;
                String s = f.toString();
                System.out.println("Text: " + s);
                System.out.println("Length: "+ s.length());
        }
}
```

- Result:

```
Text: 1.123
Length: 5
```

ArrayList

- ArrayList:
  - a class
  - a resizable array
  - package: java.util

- Syntax:

```
ArrayList<ClassName> a = new ArrayList<ClassName>()
```

- Ex:

```
ArrayList<String> fullName = new ArrayList<String>();
ArrayList<int> n = new ArrayList<int>();// error, why?
```

ArrayList

- Method:
  - add(): to add elements to the ArrayList
  - get(): to access an element
  - set(): to change an element
  - remove(): to remove an element
  - clear(): to remove all the elements
  - size(): to find out how many elements

ArrayList

- Ex:

```
import java.util.ArrayList;
public class Test {
        public static void main(String[] args) {
                ArrayList<String> animalList = new ArrayList<String>();
                animalList.add("Cat");
                animalList.add("Panther");
                animalList.add("Tiger");
                animalList.set(0,"Lion");
                for (int i = 0; i < animalList.size(); i++) {
                        System.out.println(animalList.get(i));
                }

//              for (String i : animalList) {
//                      System.out.println(i);
//              }
        }
}
```

Sort an ArrayList

- Package: java.util.ArrayList
- Package: java.util.Collections
- Collections.sort(arrayList)
- Ex:

```
import java.util.ArrayList;
import java.util.Collections;

public class Test {
   public static void main(String[] args) {
            ArrayList<String> arrayList = new ArrayList<String>();

            arrayList.add("Trâu");
            arrayList.add("Bò");
            arrayList.add("Gà");

            Collections.sort(arrayList);
            for (String i : arrayList) {
                        System.out.println(i);
            }}}
```

Lambda Expressions

- Lambda expression:
  - a short block of code
  - take in parameters and returns a value
  - similar to methods, but do not need a name
- Syntax:

  parameter**1** -> expression

  (parameter**1**, parameter**2**, parameter***n***) -> expression

  (parameter**1**, parameter**2**, parameter***n***) -> {code block;}

Lambda Expressions

- Ex:

```
import java.util.ArrayList;

public class Test {
        public static void main(String[] args) {
                ArrayList<Integer> arrayList = new ArrayList<Integer>();
                arrayList.add(3);
                arrayList.add(1);
                arrayList.add(2);
                arrayList.add(7);

                arrayList.sort((obj1, obj2) -> obj1 - obj2);

                for (int i : arrayList) {
                        System.out.println(i);
                }
        }
}
```

Lambda Expressions

- Ex:

```
import java.util.ArrayList;
import java.util.Collections;

public class Test {
        public static void main(String[] args) {

                ArrayList<String> arrayList = new ArrayList<String>();

                arrayList.add("Trâu");
                arrayList.add("Bò");
                arrayList.add("Gà");

                arrayList.sort(Comparator.comparing(o1 -> o1));

                //Collections.sort(arrayList);

                arrayList.forEach(a -> System.out.println(a));
        }}
```

Lambda Expressions

- Ex:

```
// setOnAction
button.setOnAction(new EventHandler<ActionEvent>() {
        @Override
        public void handle(ActionEvent event) {
                cs.cuaSo2(); // Open a window 2
        }
});
```

```
// setOnAction with Lambda
button.setOnAction(e -> {
        cs.cuaSo2();
});
```

File handling

FILE HANDLING

- Package: java.io.File

Method	Type	Description
canRead()	Boolean	Tests whether the file is readable or not
canWrite()	Boolean	Tests whether the file is writable or not
createNewFile()	Boolean	Creates an empty file
delete()	Boolean	Deletes a file
exists()	Boolean	Tests whether the file exists
getName()	String	Returns the name of the file
getAbsolutePath()	String	Returns the absolute pathname of the file
length()	Long	Returns the size of the file in bytes
list()	String[]	Returns an array of the files in the directory
mkdir()	Boolean	Creates a directory

Create a new file

- Ex

```
import java.io.File;
import java.io.IOException;

public class Test {
   public static void main(String[] args) {
          try {
                File file = new File("D:\\test.txt");
                if (file.createNewFile()) {
                      System.out.println("File created: " + file.getName());
                } else {
                      System.out.println("File already exists.");
                }
          } catch (IOException e) {
                System.out.println("An error occurred."+ e.getMessage());
          }
   }
}
```

Read

- Package: java.util.Scanner
- Method:
  - hasNextLine()
  - nextLine()
  - close()

```
import java.io.*;
import java.util.Scanner;

public class Test {
  public static void main(String[] args) {
    try {
      File file = new File("D:\\test.txt");
      Scanner reader = new Scanner(file);
```

```
while (reader.hasNextLine()) {
        String data = reader.nextLine();
        System.out.println(data);
      }
      reader.close();
    } catch (Exception e) {
      System.out.println("An error occurred." + e.getMessage());
    }}}
```

Write

- Package: java.util.FileWriter
- FileWriter writer = **new FileWriter**("D:\\test.txt");
- FileWriter writer = **new FileWriter**("D:\\test.txt", **true**);
- Method: write(), close()
- Ex:

```
import java.io.FileWriter;
public class Test {
  public static void main(String[] args) {
    try {
      FileWriter writer = new FileWriter("D:\\test.txt", true);

      writer.write("Con đường nho nhỏ gió xiêu xiêu\n");
      writer.write("Lả lả cành hoang nắng trở chiều\n");
      writer.write("Buổi ấy lòng ta nghe ý bạn\n");
      writer.write("Lần đầu rung động nỗi cô liêu\n");

      writer.close();
      System.out.println("Successfully wrote to the file.");
    } catch (Exception e) {
      System.out.println("An error occurred." + e.getMessage());
    }}}
```

Note

- To read and write files in Java:
  - FileInputStream
  - FileOutputStream
  - ObjectInputStream
  - ObjectOutputStream
  - FileReader
  - BufferedReader
  - BufferedWriter
  - …

Q&A