Bài 24
Xử lý ngoại lệ và thao tác với file text

Môn học: PF-JAVA

Mục tiêu

- Trình bày được cơ chế của ngoại lệ
- Sử dụng được cấu trúc try-catch-finally
- Sử dụng được các Wrapper Class
- Trình bày được gói java.io
- Trình bày được Stream
- Thực hiện được thao tác cơ bản với file và thư mục
- Thực hiện được các thao tác đọc và ghi với file text

Ngoại lệ

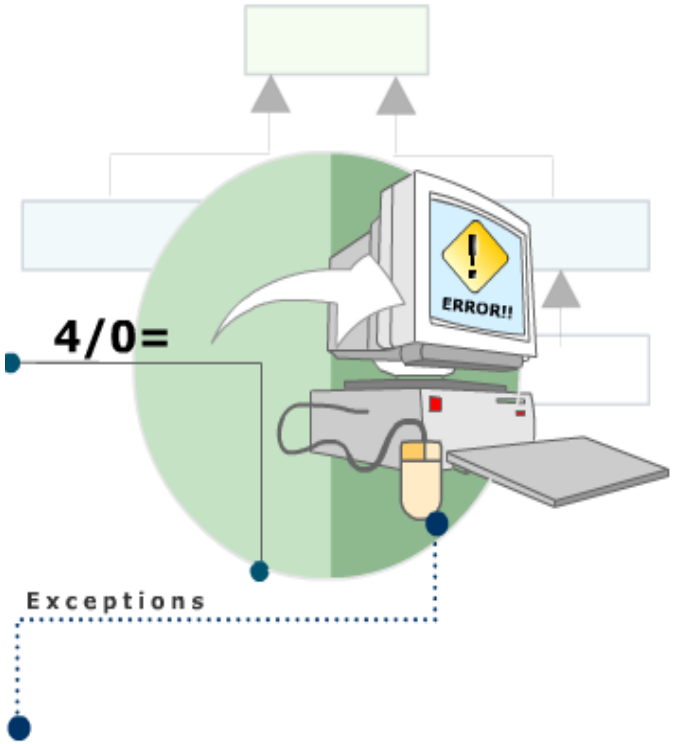
Khái niệm về Exception

Phân loại Exceptions

Xử lý ngoại lệ trong Java

Ngoại lệ

- Ngoại lệ là các lỗi phát sinh trong quá trình thực thi
- Ví dụ:
  - Lỗi khi lái xe
  - Lỗi khi máy tính xử lý phép chia cho 0



Các kiểu lỗi thường xảy ra

- Lỗi cú pháp
- Lỗi khi chạy chương trình
- Lỗi về tính logic cấu trúc của chương trình

Lỗi cú pháp

- Lỗi cú pháp (Syntax Error) xảy ra tại thời gian biên dịch trong các ngôn ngữ chương trình truyền thống và tại thời gian phiên dịch trong Javascript.
- Ví dụ, dòng sau gây ra một lỗi cú pháp bởi vì nó thiếu dấu chấm phẩy kết thúc một lệnh:

```
float area = width * height
```

Lỗi khi chạy chương trình

- Lỗi trong khi chạy chương trình (Runtime Error) xảy ra trong suốt thời gian thực thi
- Ví dụ, dòng sau tạo một Runtime Error bởi vì ở đây cú pháp là đúng, nhưng trong khi chạy, nó cố gắng gọi một phương thức mà không tồn tại.

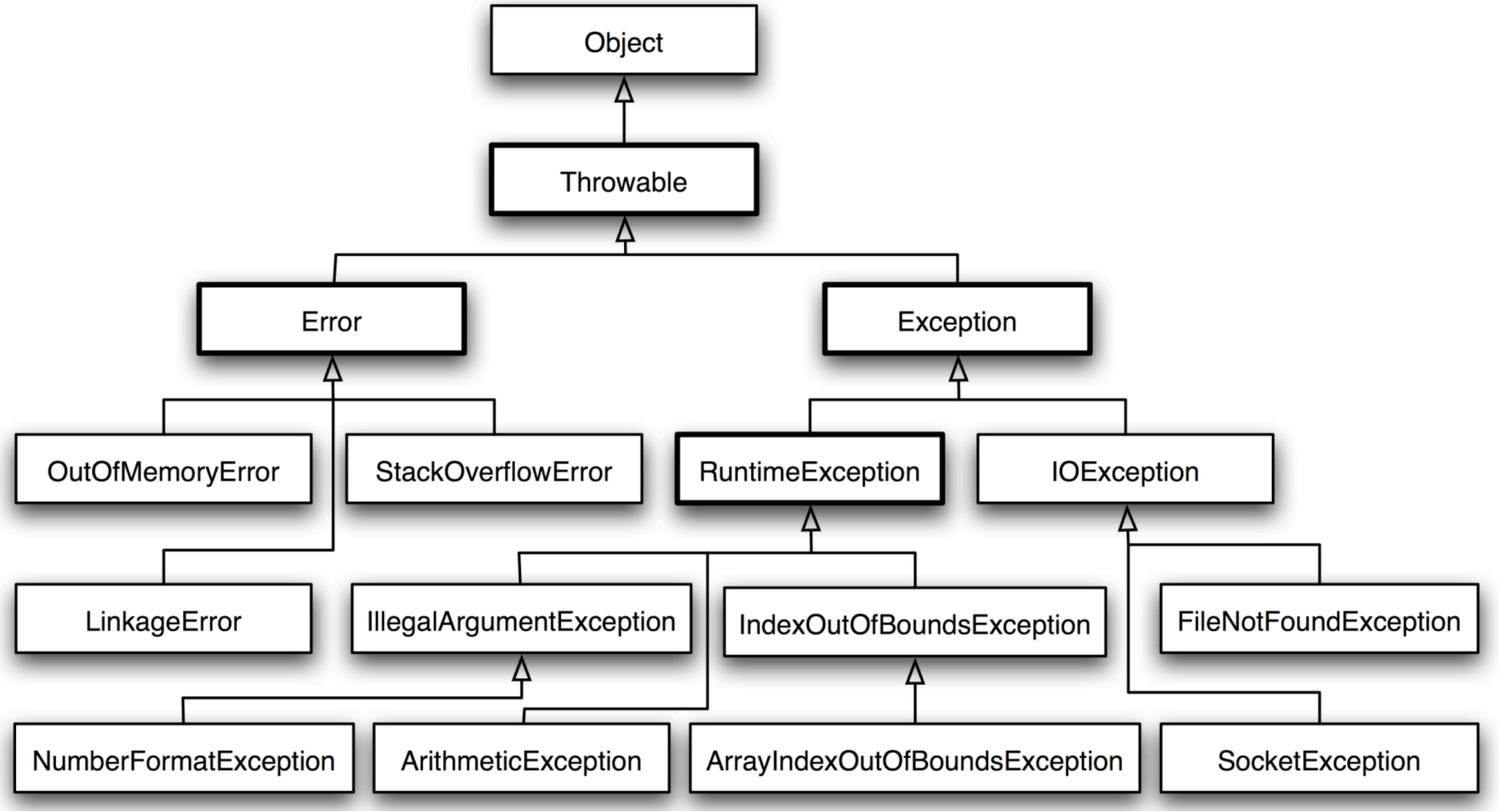
```
public static void main(String[] args) {
    int []arr = {4, 12, 7, 8, 1, 6, 9};
    int index = minValue(arr);
    System.out.println("The smallest element is: " + arr[index]);
}
```

Lỗi logic

- Lỗi về tính logic của cấu trúc chương trình (Logic Error) là kiểu lỗi khó để có thể tìm dấu vết.
- Xảy ra khi bạn tạo một lỗi về tính logic mà điều khiển script của bạn và không nhận được kết quả như mong đợi.
- Khó nắm bắt được các lỗi này, bởi vì nó phụ thuộc vào yêu cầu và kiểu logic mà bạn đặt vào chương trình.

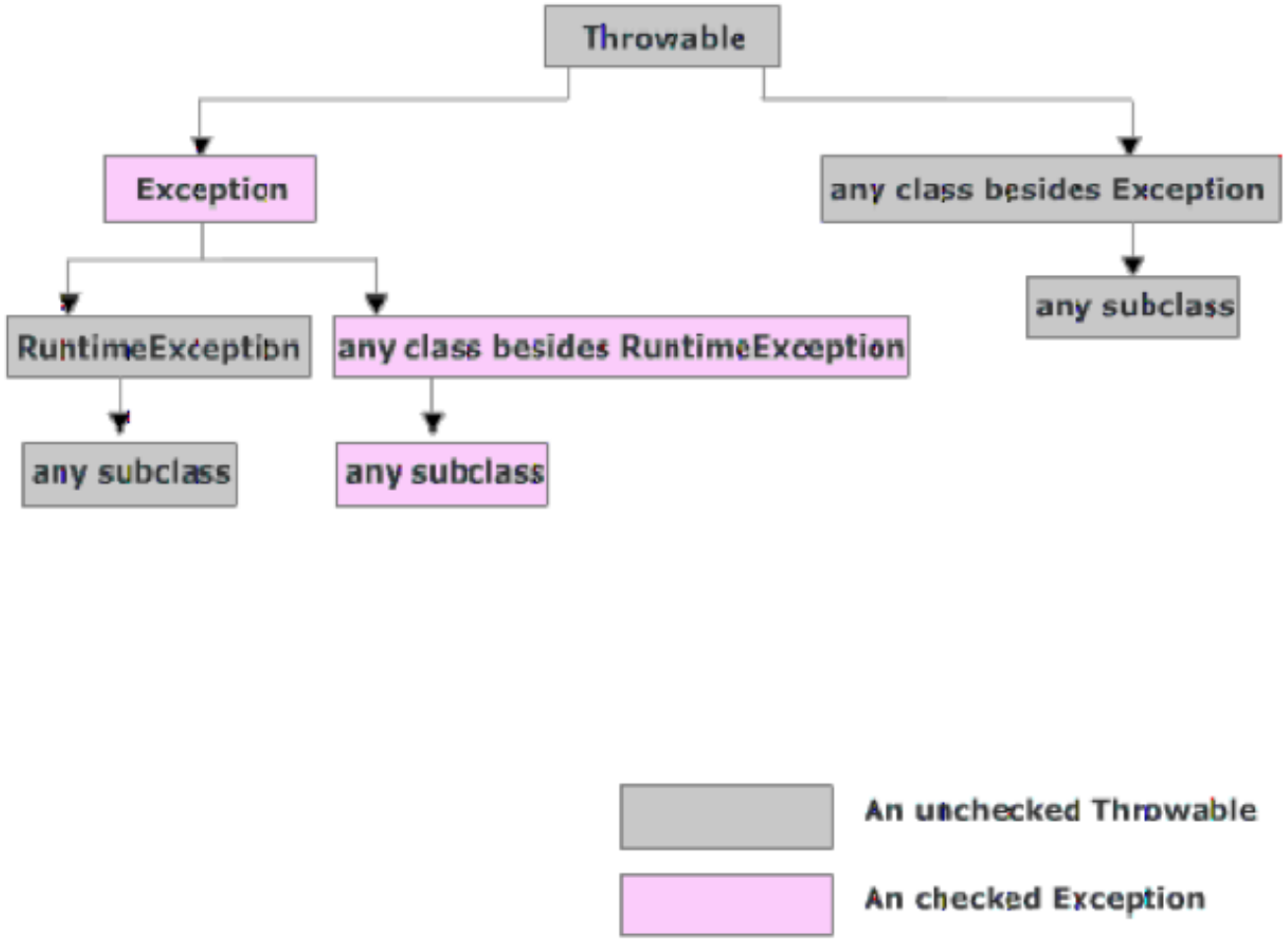
Ngoại lệ trong Java

- Lớp Throwable xử lý lỗi (error) và ngoại lệ (exception)



Phân loại ngoại lệ

- Ngoại lệ chia làm 2 loại Checked (màu nâu) và Unchecked (màu hồng)



Checked Exceptions

- Lỗi được phát sinh trong tình huống chạy chương trình thông thường.
- Ví du: Đọc một file lỗi, người dùng nhập liệu sai, lỗi mạng.
- Những Exception này cần được xử lý để tránh biên dịch lỗi. Những lỗi này được kiểm tra trong quá trình biên dịch.
- Tất cả Checked Exceptions đều được dẫn xuất từ lớp Exception.
- Ví dụ:
  - Lớp RuntimeException
  - Lớp InterruptedException
  - NoSuchMethodException

Unchecked Exceptions

- Được tạo ra trong tình huống được coi là "không thể cứu vãn" cho một chương trình (được tạo ra trong lúc đã chạy chương trình) ví dụ:
  - Cố truy cập vào một phần tử nằm ngoài phạm vi chỉ số của một mảng.
- Những ngoại lệ này không nhất thiết phải được kiểm tra trong một ứng dụng.
- Tất cả Unchecked Exceptions là dẫn xuất của lớp RuntimeException
- Ví dụ
  - Lớp ArithmeticException
  - Lớp ArrayIndexOutOfBoundsException
  - IllegalArgumentException
  - NagetiveArraySizeException
  - NullPointerException
  - NumberFormatException
  - StringIndexOutOfBoundsException

Xử lý ngoại lệ trong Java

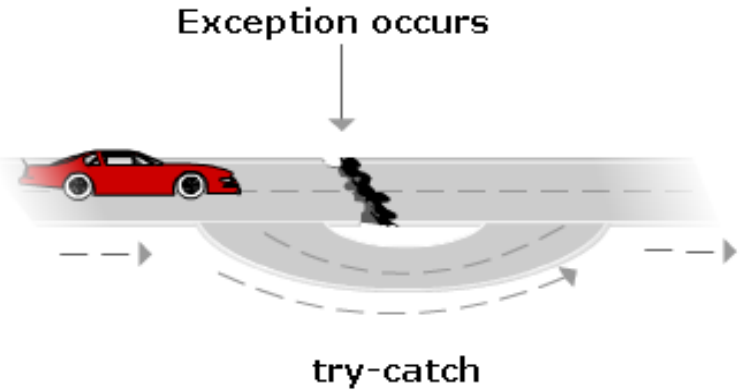
- Sử dụng *try-catch*
- Sử dụng *finally*
- Sử dụng từ khóa *throw* và *throws*
- Sử dụng nhiều khối *catch*

Sử dụng khối try-catch

- Khối try-catch dùng để
  - Tách phần giải quyết lỗi ra khỏi phần có thể sinh lỗi
  - Quy định các loại ngoại lệ được bắt tại mức thực thi hiện hành
- Cú pháp

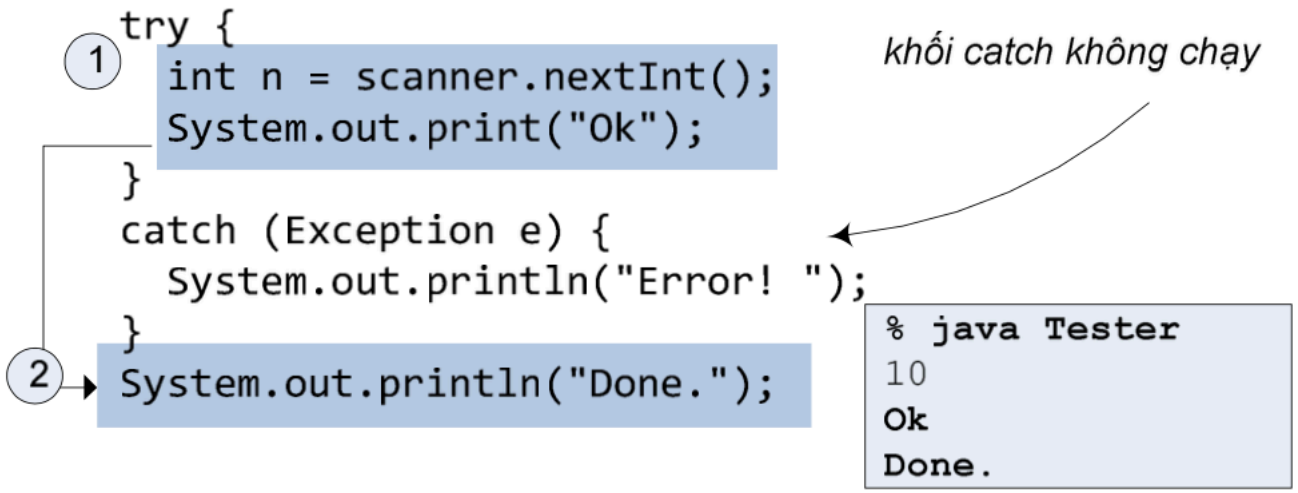
```
try {
        Statement_1;
        Statement_2;
}
catch(ExceptionType  ObjectName){
        Statement_1;
}
```

  - Mã liên quan đến thuật toán nằm trong khối try
  - Mã giải quyết lỗi đặt trong các khối catch



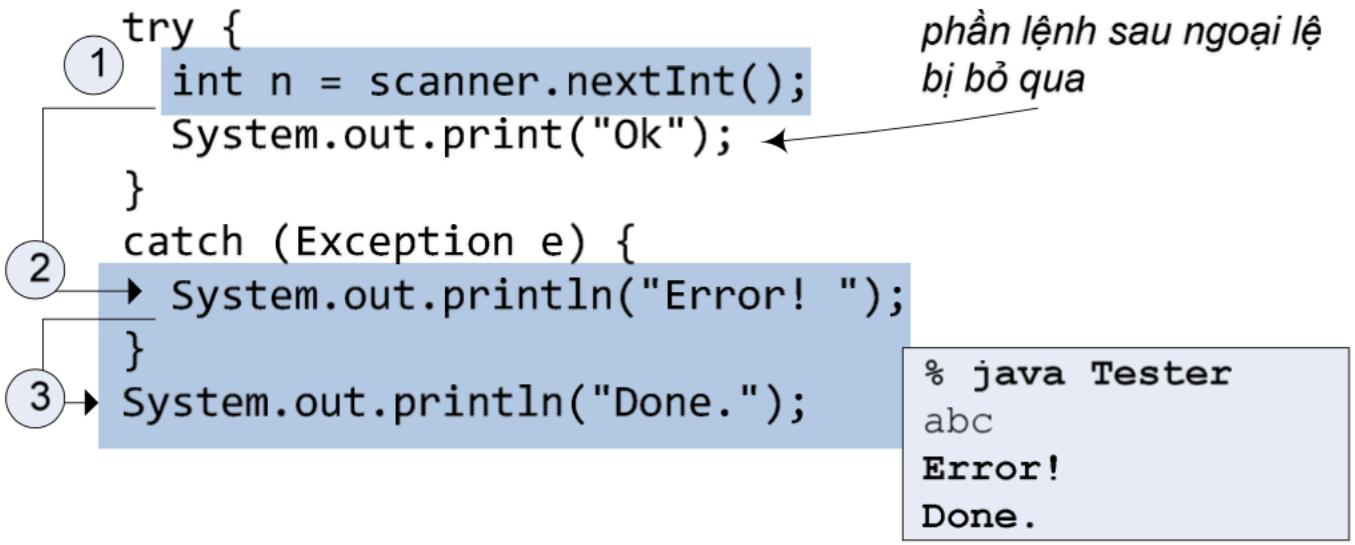
Sử dụng khối try-catch

- Phương thức được gọi thành công và khối try được thực thi đầy đủ cho đến lệnh cuối cùng, còn khối catch bị bỏ qua vì không có ngoại lệ nào phải xử lý.



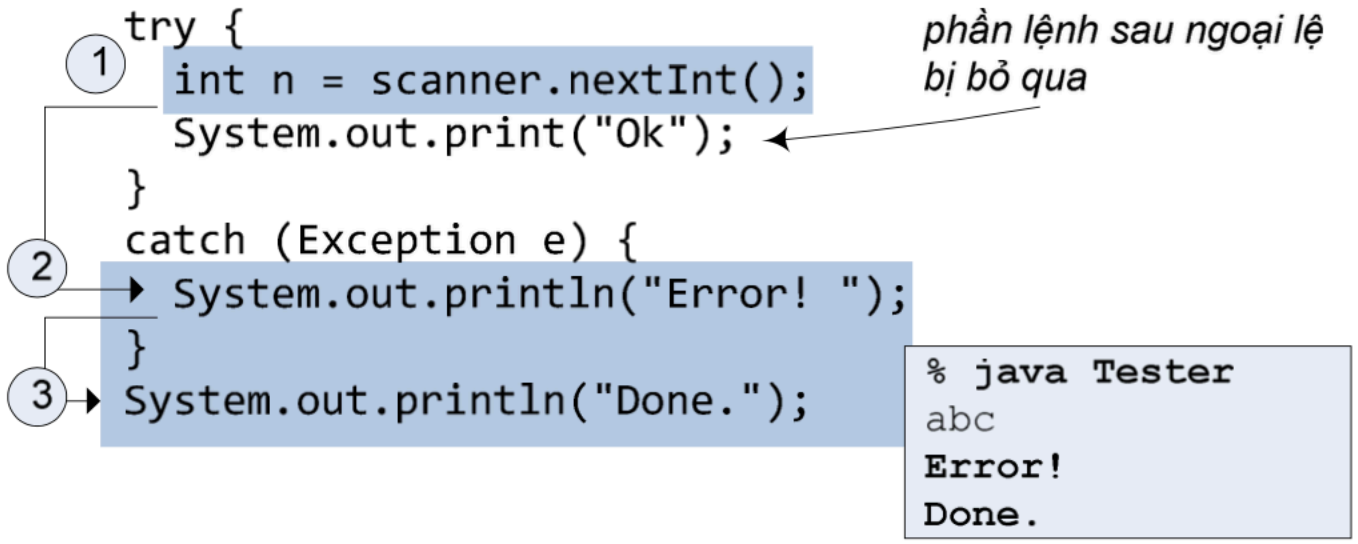
Sử dụng khối try-catch

- Phương thức được gọi ném ngoại lệ và khối catch bắt được ngoại lệ đó



Sử dụng khối try-catch

- Phương thức được gọi ném ngoại lệ nhưng khối catch không bắt được ngoại lệ

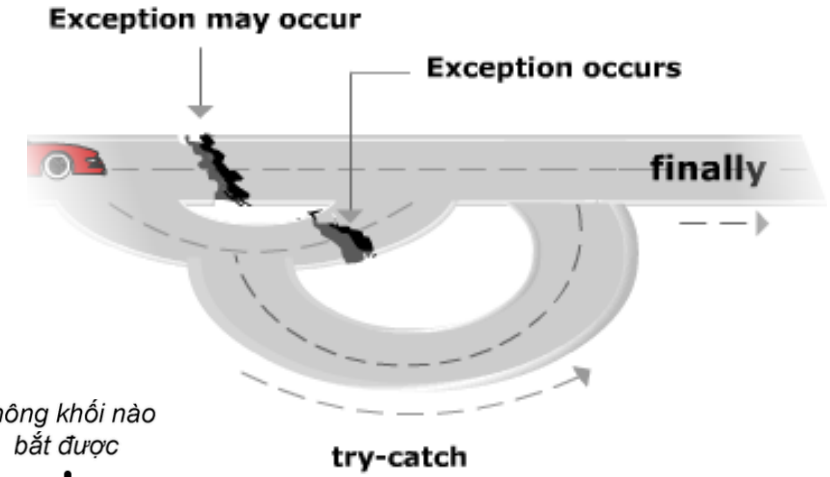


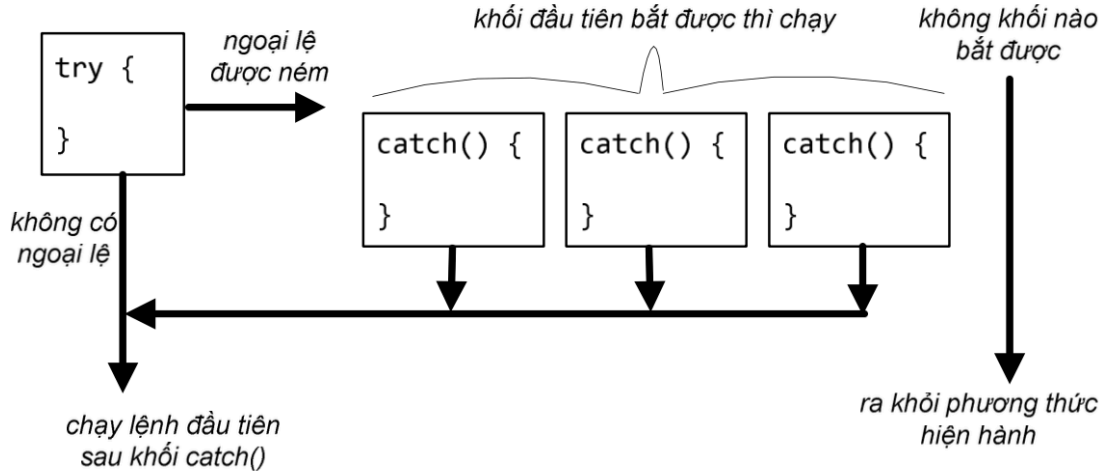
Sử dụng nhiều khối catch

- Khối mã try có thể có nhiều ngoại lệ xảy ra. Sử dụng nhiều khối catch để bắt và xử lý chi tiết các ngoại lệ đó.

Cú pháp

```
try{
}catch(ExceptionType name){
}catch(ExceptionType name){
}
```





Sử dụng nhiều khối catch

- Ví dụ

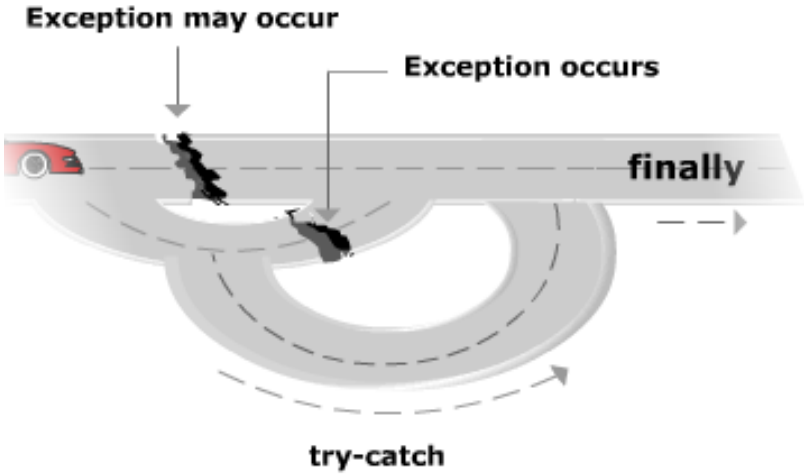
```
try{
 int a[]=new int[5];
 a[5]=30/0;
}
catch(ArithmeticException e){System.out.println("task1 is completed");}
catch(ArrayIndexOutOfBoundsException e){System.out.println("task 2 completed");}
catch(Exception e){System.out.println("common task completed");}
```

- Lưu ý
  - Đối tượng trong khối catch sau phải cùng cấp hoặc có cấp lớn lớn hơn đối tượng thuộc catch trước.
  - Nếu không biết đoạn mã lệnh có khả năng gây lỗi gì thì sử dụng lớp Exception để bắt.

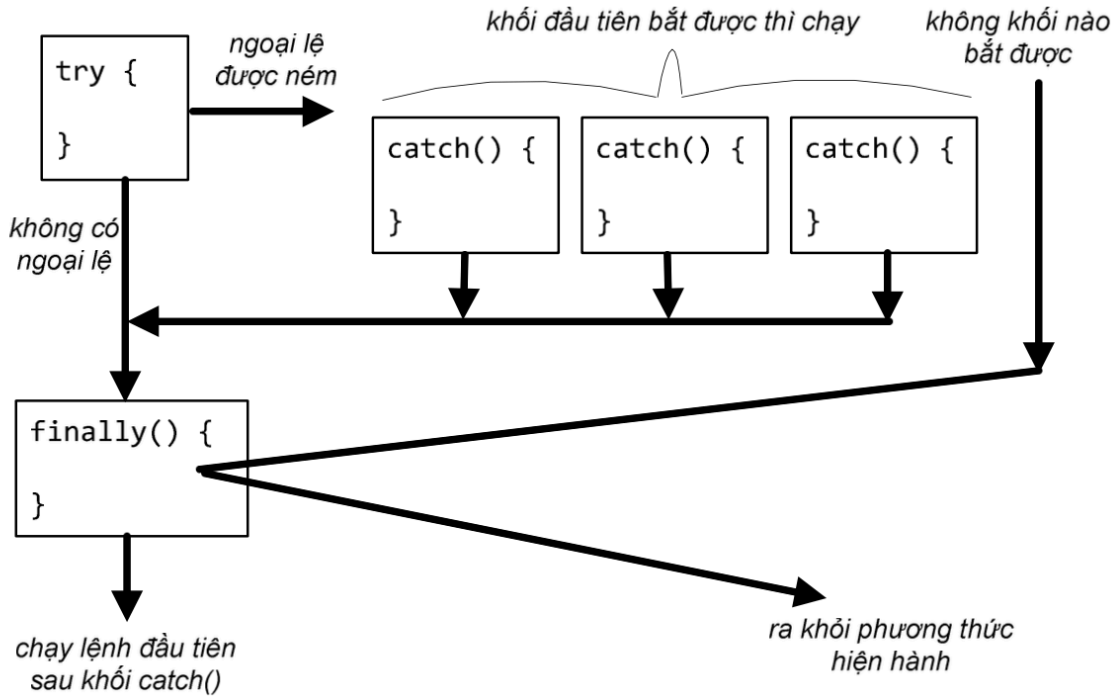
Khối finally

- Khối **finally** mà sẽ luôn luôn thực thi vô điều kiện sau **try/catch**.
- Cú pháp:

```
try {
        Statement_1;
        Statement_2;
}
catch(ExceptionType  ObjectName){
        Statement_1;
}
finally{
        //Clean up code
        Statement_1;
}
```



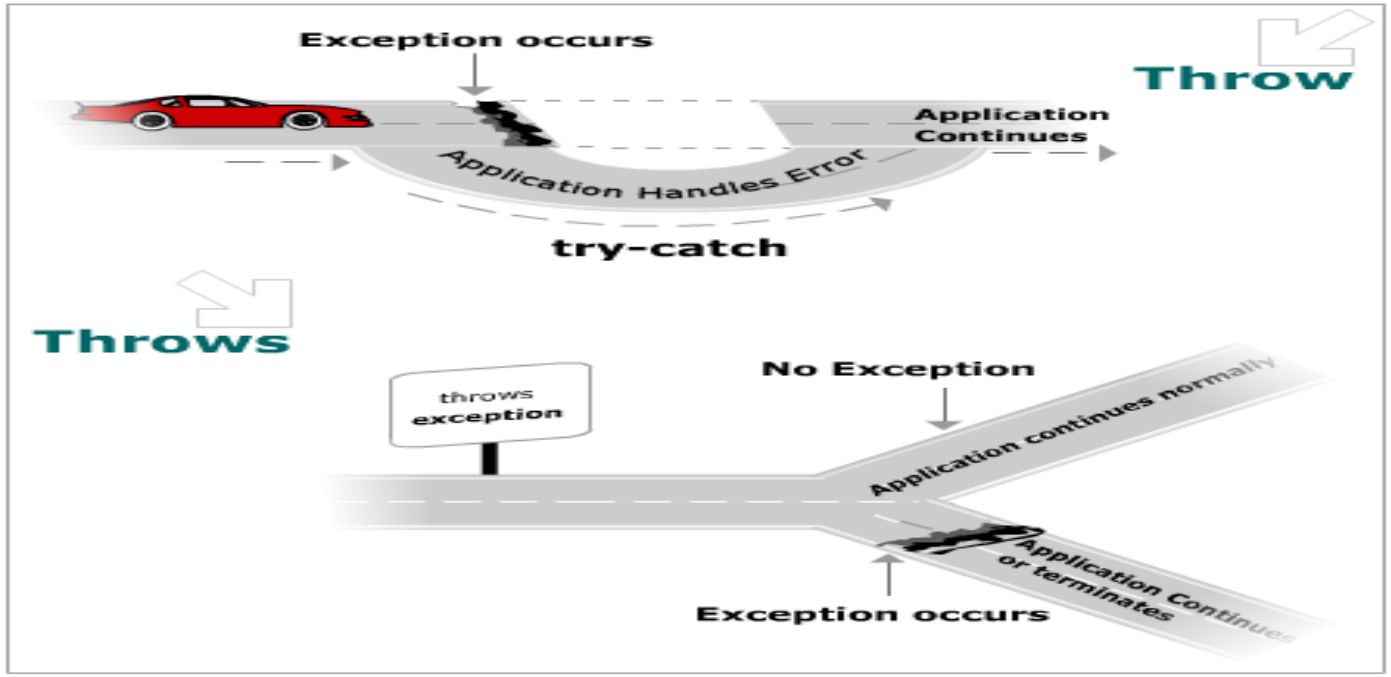
Khối finally



Lưu ý, về mặt cú pháp, ta không thể chèn mã vào giữa các phần try..catch và finally trong một khối try/catch. Khối try thì bắt buộc phải có, nhưng các khối catch và finally thì không, tuy nhiên sau một khối try phải có ít nhất một khối catch hoặc finally.

Ném ngoại lệ

- Sử dụng lệnh **throw** để ném các lỗi Runtime Error có sẵn hoặc các lỗi do người dùng tự định nghĩ. Sau đó các lỗi này có thể được bắt và bạn có thể thực hiện một hành động hợp lý.



Sử dụng từ khóa *throw* và *throws*

- Câu lệnh **throw** được dùng khi cần phát sinh ngoại lệ trong một phương thức.
- Từ khóa **throws** được dùng với phương thức để gây ra bất cứ checked hay unchecked exception nào và đồng thời nhường công việc xử lý ngoại lệ đó cho người gọi phương thức này.
- Ngoại trừ những Error hoặc RuntimeException và các subclasses của chúng, sử dụng throws là cần thiết cho tất cả các ngoại lệ.

Ví dụ ném và bắt ngoại lệ

```
public class Fraction {
  private int numerator, denominator;

  public Fraction (int n, int d) throws Exception {
    if (d==0) throw new Exception();

    numerator = n;
    denominator = d;
  }
}
```

tuyên bố rằng phương thức này có thể ném ngoại lệ loại Exception

tạo một đối tượng Exception mới và ném nó tới nơi gọi phương thức

```
public class TestFraction {
  public static void main(String [] args) {
    try {
      Fraction f = new Fraction (2,0);
    }
    catch (Exception e) {
      e.printStackTrace();
    }
  }
}
```

nếu không khắc phục được sự cố thì ít nhất cũng in ra thông tin lần vết sự cố

Né ngoại lệ

- Né ngoại lệ là cơ chế khi một phương thức dùng đến những lời gọi phương thức có thể phát sinh ngoại lệ, nhưng người dùng không muốn xử lý tại phương thức đó.

```
public class FileWriter {
  public static void write(String fileName, String s)
                                  throws FileNotFoundException {
    File file = new File(fileName);
    PrintWriter out = new PrintWriter(file);

    out.println(s);
    out.close();
  }
}
```

né ngoại lệ do new PrintWriter ném bằng cách dùng khai báo throws, Ngoại lệ không được bắt sẽ rơi ra ngoài tới nơi gọi phương thức này

```
public class WriteToFile {
  public static void main(String[] args) {
    try {
      FileWriter.write("hello.txt", "Hello!");
    }
    catch (Exception e) {
      e.printStackTrace();
    }
  }
}
```

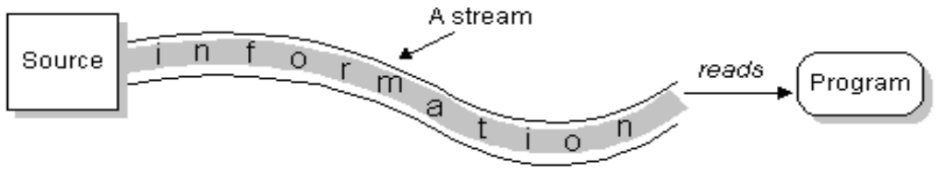
bắt và xử lý ngoại lệ được ném ra từ trong FileWriter.write

Thao tác với file text

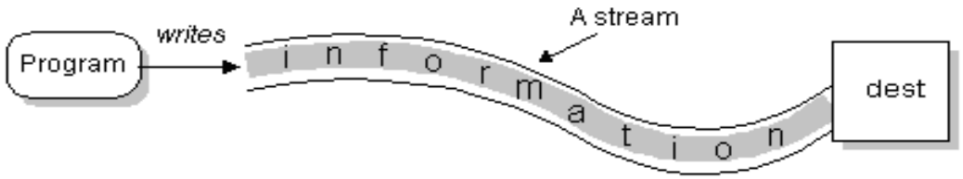
Giới thiệu gói java.io
Giới thiệu về Stream
Thao tác cơ bản với file và thư mục
Thao tác đọc và ghi với file text

Giới thiệu về Stream

- Các hoạt động nhập/xuất dữ liệu như nhập dữ liệu từ bàn phím, đọc dữ liệu từ file, ghi dữ liệu màn hình, ghi ra file, ghi ra đĩa, ghi ra máy in ... được gọi là stream (dòng).
- Dòng vào là luồng cho phép chương trình đọc dữ liệu từ một nguồn nào đó như bàn phím, file, máy scan ...



- Dòng ra là luồng cho phép chương trình ghi dữ liệu lên nó để chuyển đến đích nào đó: màn hình, file, máy in ...

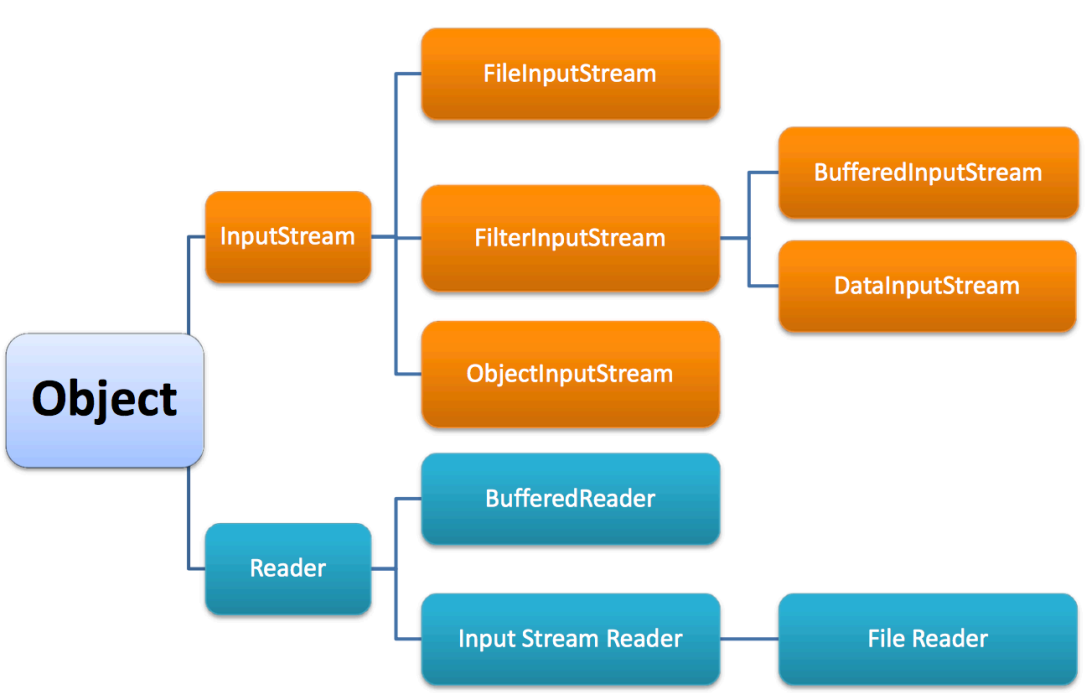


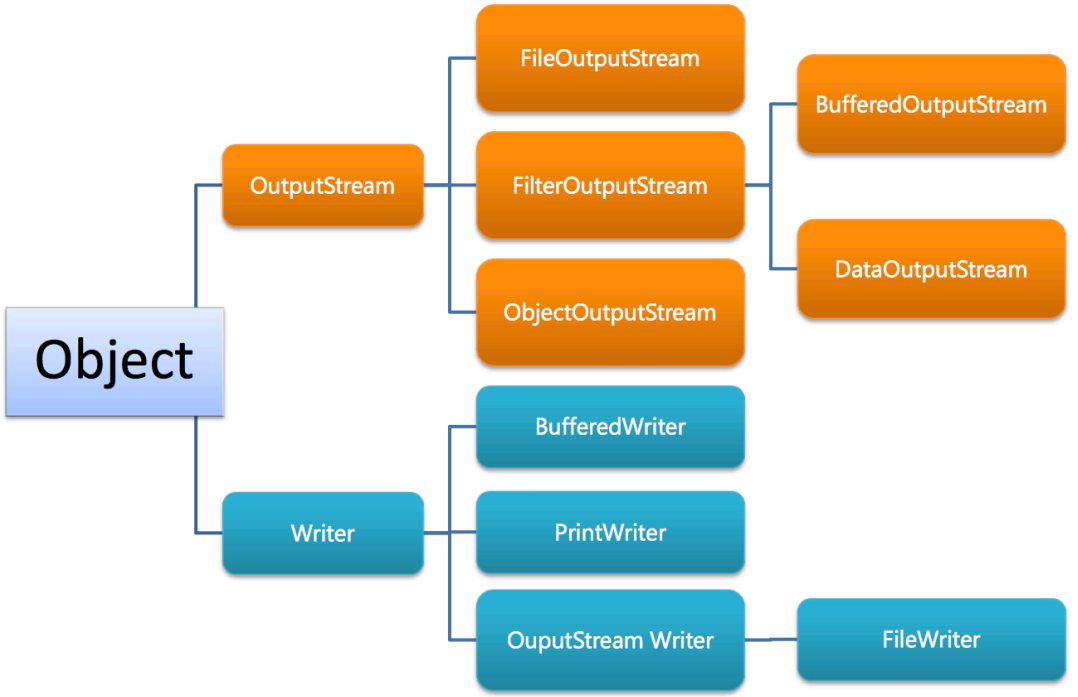
Các loại luồng dữ liệu

- Có 2 kiểu luồng trong Java: dòng bye (dòng nhị phân) và dòng character (dòng văn bản)
- Dòng byte (byte - based stream)
  - Hỗ trợ việc xuất nhập dữ liệu theo byte
  - Thường được dùng khi đọc ghi dữ liệu nhị phân
- Dòng character (character - based stream)
  - Luồng character được thiết kế hỗ trợ việc xuất nhập dữ liệu kiểu ký tự

Giới thiệu gói java.io

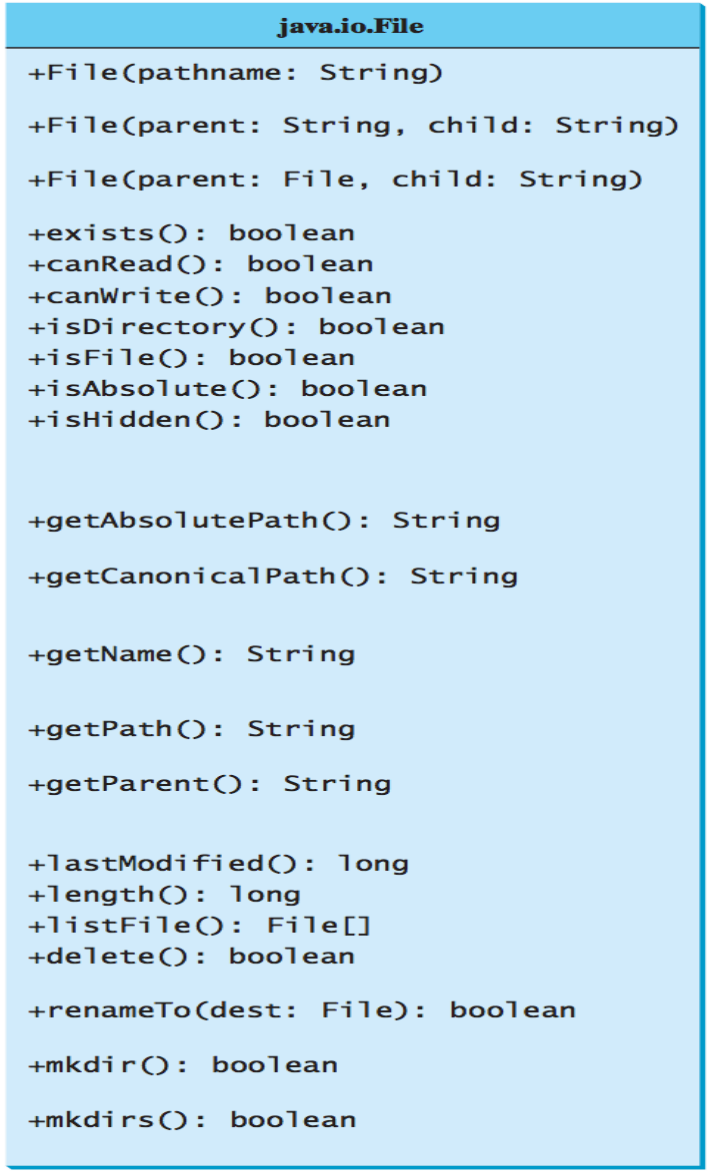
- Gói java.io chứa các interface, class để thao tác với dòng dữ liệu.





Lớp File trong java

- Lớp File thuộc gói java.io đại diện cho một file hoặc một thư mục.
- Lớp này không có các tiện ích đọc/ghi file, nhưng nó là đại diện an toàn cho file hơn là chuỗi ký tự tên file.
- Hầu hết các lớp lấy tên file làm tham số cho phương thức khởi tạo.



Lớp File trong java

- Tạo một đối tượng File đại diện cho một file đang tồn tại

  File f = new File(“foo.txt”);

- Tạo một thư mục mới

  File dir = new File(“Books”);

  dir.mkdir();

- Liệt kê nội dung của một thư mục

```
if (dir.isDirectory()) {
  String[] dirContents = dir.list();
  for (int i = 0; i < dirContents; i++)
    System.out.println(dirContents[i]);
}
```

Lớp File trong java

- Lấy đường dẫn tuyệt đối của file hoặc thư mục

```
System.out.println(dir.getAbsolutePath());
```

- Xoá file hoặc thư mục

```
boolean isDeleted = f.delete();
```

Ví dụ sử dụng lớp File

```
public class TestFileClass {
  public static void main(String[] args) {
    java.io.File file = new java.io.File("image/us.gif");                  create a File
    System.out.println("Does it exist? " + file.exists());                 exists()
    System.out.println("The file has " + file.length() + " bytes");        length()
    System.out.println("Can it be read? " + file.canRead());               canRead()
    System.out.println("Can it be written? " + file.canWrite());           canWrite()
    System.out.println("Is it a directory? " + file.isDirectory());        isDirectory()
    System.out.println("Is it a file? " + file.isFile());                  isFile()
    System.out.println("Is it absolute? " + file.isAbsolute());            isAbsolute()
    System.out.println("Is it hidden? " + file.isHidden());                isHidden()
    System.out.println("Absolute path is " +
      file.getAbsolutePath());                                             getAbsolutePath()
    System.out.println("Last modified on " +
      new java.util.Date(file.lastModified()));                            lastModified()
  }
}
```

Thao tác ghi với file text

- Sử dụng lớp FileWriter thực hiện ghi chuỗi ký tư ra file text

```
import java.io.*;

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

    try {
      FileWriter writer = new FileWriter("Hello.txt");

      writer.write("Hello!");

      writer.close();

    }
    catch (IOException e) {
      e.printStackTrace();
    }
  }
}
```

import gói java.io để dùng FileWriter

mở file để ghi

các lệnh I/O đều có thể gây ngoại lệ nên phải có try/catch

write() ghi một đối tượng String ra file

đóng file khi xong việc

Bộ nhớ đệm

- Bộ nhớ đệm (buffer) cho một nơi lưu trữ tạm thời để tăng hiệu quả của thao tác đọc/ghi đữ liệu
- Bộ nhớ đệm trong java được hỗ trợ qua việc sử dụng lớp BufferWriter trong gói java.io
- Cách sử dụng BufferWriter:

```
BufferWriter writer = new BufferWriter(new FileWriter(aFile));
```

Thao tác đọc file text

- Sử dụng lớp FileReader thực hiện đọc file text và BufferReader để tăng hiệu quả đọc.

```
import java.io.*;

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

    try {
      File inFile = new File("Hello.txt");
      FileReader fileReader = new FileReader(inFile);

      BufferReader reader = new BufferReader(fileReader);

      String line = null;

      while ((line = reader.readLine()) != null) {
        System.out.println(line);
      }
      reader.close();

    }
    catch (Exception e) {
      e.printStackTrace();
    }
  }
}
```

nối FileReader với một file text

nối BufferReader với FileReader

đọc từng dòng cho đến khi không đọc được gì nữa (hết file)

Tổng kết

- Ngoại lệ là các lỗi phát sinh trong quá trình thực thi
- Xử lý ngoại lệ với khối try-catch-finally
- Sử dụng lệnh **throw** để ném các lỗi
- Các hoạt động nhập/xuất dữ liệu như nhập dữ liệu từ bàn phím, đọc dữ liệu từ file, ghi dữ liệu màn hình, ghi ra file, ghi ra đĩa, ghi ra máy in ...được gọi là stream (dòng).
- Có 2 kiểu luồng trong Java: dòng bye (dòng nhị phân) và dòng character (dòng văn bản)
- Lớp File thuộc gói java.io đại diện cho một file hoặc một thư mục.
- Sử dụng lớp FileWriter thực hiện ghi chuỗi ký tư ra file text
- Sử dụng lớp FileReader thực hiện đọc file text và BufferReader để tăng hiệu quả đọc.