វិទ្យាស្ថានជាតិសហគ្រិនភាព និងនវានុវត្តន៍

JAVA Programming
Group 5

1. ខុម មករា
2. ឈឹម សិនន
3. ប៊ិន ភក្តី

JAVA I/O

Java I/O (Input and Output) : ប្រើសម្រាប់ process the input and output
និងអាចតភ្ជាប់ទៅនឹងឧបករណ៍តាមរយៈប្រព័ន្ធ I/O របស់Java គ្រប់ streams
ទាំងអស់ធ្វើសកម្មភាពក្នុងលក្ខណៈដូចគ្នា ទោះជាឧបករណ៍ដែលវាតភ្ជាប់ទៅនោះមានលក្ខណៈផ្សេងៗគ្

Ex.methodsដូចគ្នាដែលយើងប្រើសំរាប់សរសេរបង្ហាញទៅលើ console
អាចប្រើសំរាប់សរសេរទៅក្នុង file ផងដែរ ។

Streams

- Byte streams ត្រូវបានកំណត់ទ្បើងដោយប្រើលំដាប់ថ្នាក់ class ចំនួនពីរៗclass ទាំងពីរដែលមានលក្ខណៈជា abstract classនោះគឺ input Stream និងOutput Stream ។
- Input Stream មានលក្ខណៈទូទៅសំរាប់ streams ដែលបញ្ចូលទិន្នន័យជា byte.
- Output Stream មានលក្ខណៈសំរាប់ streams ដែលបញ្ចេញជា byte ។

Character Stream Classes

- Character streams ត្រូវបានកំណត់ឡើងដោយប្រើលំដាប់ថ្នាក់ class ចំនួនពីរ។class ទាំងពីរដែលមានលក្ខណៈជា abstract class នោះគឺ Reader និង Writer ។
  - Reader ត្រូវបានប្រើសំរាប់បញ្ចូលទិន្នន័យ .
  - Writer ត្រូវបានប្រើសំរាប់បញ្ចេញទិន្នន័យ .
- classes ដែលបានមកពី Reader និង Writerធ្វើប្រតិបត្តិការនៅលើ streamនៃតួអក្សរ Unicode ។

ការប្រើByte Streams

- ★ byte streams គឺជា Input Streamនិង Output Stream classes ។
- ★ System.in គឺមានលក្ខណៈជា objects Input Stream ដែលអាចអោយយើងចូលទៅប្រើ methods កំណត់ដោយ Input Stream ។
- ★ Input Stream មាន method តែមួយគត់គឺសំរាប់បញ្ជូលជា bytes នោះ គឺ read() ។
- ★ វាមានទម្រង់:
  - int read() throws IO Exception. (ប្រើសម្រាប់អានតួអក្សរ)
  - int read(bytedata[]) throws IO Exception. (អានជា Bytes តាមការបញ្ជូលstream ហើយដាក់ទៅក្នុងData រហូតពេញArray)
  - int read(bytedata[], intstart, intmax) throws IO Exception. (បញ្ជូលនិងផ្ទុកទិន្នន័យទៅដាក់ក្នុងData).

ការបញ្ចេញទិន្នន័យទៅលើ file

- ★ ដើម្បីបើក file មួយបង្ហាញមកក្រៅ គេត្រូវតែបង្កើត object_is_FileOutputStream ។
  - FileOutputStream(String fileName) throws FileNotFoundException
  - FileOutputStream(String fileName, boolean append) throws FileNotFoundException
- ★ ទំរង់ទីមួយ កាលណា file មួយដែលត្រូវបង្ហាញមកក្រៅបានបើកឡើងដោយមានឈ្មោះដូច file ដែលមានរួចហើយនោះ វានឹងធ្វើអោយបាត់បង់នូវ fileចាស់។
- ★ ទំរង់ទី២ បើសិន append មានតំលៃ true នោះលទ្ធផលបង្ហាញមកក្រៅត្រូវតភ្ជាប់នៅខាងចុងនៃ file ផ្ទុយទៅវិញ file នឹងត្រូវសរសេរជាន់បាត់ខ្លឹមសារដែលមានពីមុន ។

ការអាន និងបង្ហាញទិន្នន័យប្រភេទ binary

- ដើម្បីអានបញ្ចូល និងសរសេរបង្ហាញនូវតំលៃ binary នៃប្រភេទទិន្នន័យទំរង់ងាយ គេប្រើ
  - Data inputStream
  - Data OutputStream
- Data Output Stream អនុវត្តតាម interface ដែលមានឈ្មោះData Output។
- interfaceនេះមានmethodសំរាប់សរសេរបង្ហាញនូវប្រភេទទិន្នន័យទំរង់ងាយទៅលើ file ។

ការប្រើ RandomAcess File

★ RandomAccessFile class បើ read() និង write() ។ វាបានអនុវត្តតាមinterfacesឈ្មោះ

- DataInput
- DataOutput

ដែលអាចអោយគេប្រើ readInt() និង writeDouble() សំរាប់អាន និងសរសេរនូវប្រភេទទិន្នន័យទំរង់ងាយ ។

★ ឧទាហរណ៍ខាងក្រោមនេះបង្ហាញនូវការបញ្ចេញ-បញ្ចូលទិន្នន័យមិនតាមលំដាប់ះ

```
// Demonstrate random access files.
import java.io.*;

class RandomAccessDemo {
  public static void main(String args[]) throws IOException {

    double data[] = { 19.4, 10.1, 123.54, 33.0, 87.9, 74.25 };
    double d;
    RandomAccessFile raf;

    try {
      raf = new RandomAccessFile("random.dat", "rw");
    } catch (FileNotFoundException exc) {
      System.out.println("Cannot open file.");
      return;
    }
    // Write values to the file.
    for (int i=0; i < data.length; i++) {
      try {
         raf.writeDouble(data[i]);
      } catch (IOException exc) {
         System.out.println("Error writing to file.");
         return;
      }
    }
```

```
try {
    // Now, read back specific values
    raf.seek(0);  // seek to first double
    d = raf.readDouble();
    System.out.println("First value is " + d);

    raf.seek(8);  // seek to second double
    d = raf.readDouble();
    System.out.println("Second value is " + d);

    raf.seek(8 * 3);    // seek to fourth double
    d = raf.readDouble();
    System.out.println("Fourth value is " + d);

    System.out.println();

    // Now, read every other value.
    System.out.println("Here is every other value: ");
```

```
        for (int i=0; i <data.length; i += 2) {
          raf.seek(8 * i);          // seek to ith double
          d = raf.readDouble();
          System.out.print(d + " ");
        }
      } catch (IOException exc) {
        System.out.println("Error seeking or reading.");
      }
      raf.close();
    }
}
```

ការប្រើ listFiles()

- ★ Java បានបន្ថែមភាពខុសគ្នាទៅអោយ list() ហៅថា listFiles() ដែលយើងអាចរកឃើញនូវសារៈសំខាន់របស់វា។

- ★ ទំរង់នៃ listFiles() បានបង្ហាញដូចខាងក្រោមនេះ
  - File[] listFiles()
  - File [] listFiles (Filename Filter FFObj)
  - File [] listFiles (FileFilter FObj)

- ★ ទំរង់ទីមួយ method អោយតំលៃជា files ទាំងអស់។
- ★ ទំរង់ទីពីរអោយតំលៃជា fileទាំងឡាយណា
  ដែលត្រូវគ្នាតាមការកំណត់របស់ FilenameFilter។
- ★ ទំរង់ទីបី នៃ listFiles() អោយតំលៃជា files ជាមួយឈ្មោះទីតាំង ដែលបំពេញអោយ
  FileFilter
  តាមការកំណត់។
- ★ FileFilter មាន method តែមួយគត់គឺ accept() ។
  ទំរង់ទូទៅរបស់វា គឺ៖
  boolean accept(Filepath)

```
import java.io.*;

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

      String dirname = "c:/j2sdk14/bin/";
      File f1 = new File(dirname);

      OnlyExt only = new OnlyExt("java");
      File f[] = f1.listFiles(only);

      for (int i=0; i < f.length; i++) {
         if (only.accept(f[i]))
            System.out.println( f[i]);
      }
   }
}
class OnlyExt implements FileFilter {
   String ext;
   public OnlyExt(String ext) {
      this.ext = "." + ext;
   }
   public boolean accept(File file) {
      return file.getAbsolutePath().endsWith(ext);
   }
}
```