COMP 102.2x
Introduction to Java Programming – Part 2
Lecture 2

T.C. Pong
Department of Computer Science & Engineering
HKUST

香港科技大學
THE HONG KONG UNIVERSITY OF SCIENCE AND TECHNOLOGY

COMP102x Part2: Lecture 2

- Character Strings
  - String class
  - String manipulation
- File Input / Output (I/O)

Your First Java Program

```
// a simple program
public class HelloWorld
{
   public static void main(String[] args)
   {
      System.out.println("Hello world!");
   }
}
```

A character string

```
// "Hello work!" is a string literal.
// The following would produce the same effect
//
// String greeting = "Hello world!";
// System.out.println(greeting);
```

comp102x.IO

```
import comp102x.IO;

// a simple program
public class HelloWorld
{
    public static void main(String[] args)
    {
        IO.outputln("Hello world!");
    }
}
```

String Manipulation

- String – a sequence of characters
- The class String includes methods for
  - examining individual characters
  - comparing strings
  - searching strings
  - extracting substrings
  - converting strings to uppercase or lowercase
- Strings are immutable – once a string object is created, it cannot be changed in place.

Char - Character type

- A data type which is used to store exactly ONE character
  - char date type is a 16-bit Unicode character
  - Note: A pair of single quote ‘ ’ is used

```
public class Students {

   /* Instance variables  */

   private char gender = 'M'; // an example of character

}
```

Char - Character type

- A compilation error occurs if we type more than one character inside a pair of single quotes ‘ ’

```
public class Students {

   /* Instance variables */

  // A compilation error occurs if more than one characters
  // are assigned to a char variable
  private char gender = 'Male';
}
```

Char - Character type

- A compilation error occurs if we type more than one character inside a pair of single quotes ‘ ’

```
public class Students {

   /* Instance variables */

  // A compilation error occurs if more than one characters
  // are assigned to a char variable
   private String gender = “Male”;
}
```

Escape sequence

- Escape sequence is for special characters
  - An extra backslash is added in front of the special character

```
// Examples of escape sequence
char singleQuote = '\'';
char doubleQuote = '\"';
char tabChar = '\t';
char nextLineChar = '\n';
char backSlashChar = '\\';
```

- IO.outputln("She received an \"A\" in COMP1022P.");

  would print out: She received an "A" in COMP1022P.

String

- What happens if more than ONE character needs to be stored?
  - Examples: Record the student's name, student ID, etc.

- Solution #1: An array of characters
  - Example:
    - char[ ] studentName;
    - char[ ] nameArray = {'M', 'a', 'r', 't', 'i', 'n' };
      String nameString = new String(nameArray);

- Solution #2: String
  - String is an abstract data type (ADT)
  - Example:
    - String studentName;

String operations

- String is an Abstract Data Type (ADT)
- In this course, some of the methods that we will discuss include:
  - length()
    - Determines the length of a string
  - charAt(int index)
    - Returns a character located at the given index
  - substring(int i, int j)
    - Returns substring with character from index i to j-1
  - equals(String other)
    - Compares the current string with another string
  - String comparison
  - String concatenation
  - …

String

- String stores a sequence of characters
  - A string stores characters from index 0 to N-1, where N is the length of the string
  - Use a pair of double quotes “ ”

```
public class Students {
    // An instance variable to store the name
    private String studentName = “Martin";
}
```

Syntax:

String *nameOfVariable* = “String message”;

String – Reference Type

- In the previous example, **studentName** is a reference variable to a string
- Consider the following example:

```
public class Students {
    // The variable can be used to reference any string
    private String studentName ;
    public Students(String stdName) {
        studentName = stdName;
    }
}
```

Length of a string

- A method length() is used to count the number of characters in a string
- Example:
  - String studentName = “Martin”;
  - int size = studentName.length();
    - The value of the variable size will be set to 6 because “Martin” has exactly 6 characters.

Accessing a character in a String

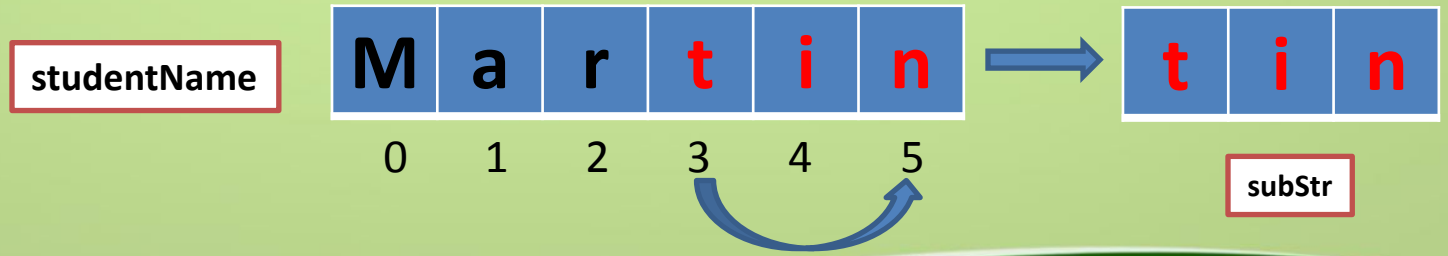
- Unlike Array, you don't need the [ ] operator
- The charAt(int index) method can be used to access a character from a string
- Example:
  - String studentName = "Martin";
  - char firstChar = studentName.charAt(0);

Getting a substring

- String substring(int i, int j)
  - A substring includes characters starting from index i to index j-1
- Example:
  - String studentName = “Martin”;
  - String subStr = studentName.substring(0, 3);

Getting a substring

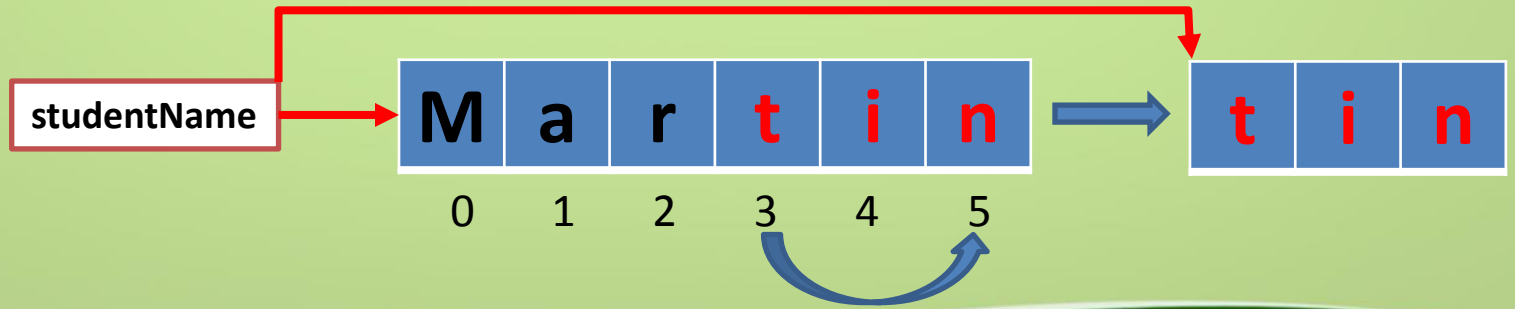
- String substring(int i)
  - A substring includes characters starting from index i to the end of the original string
- Example:
  - String studentName = “Martin”;
  - String subStr = studentName.substring(3);



Getting a substring

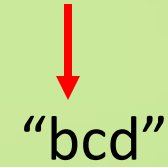
- Strings are immutable
  - substring doesn't change the original string
- Example:

```
String studentName = "Martin";
studentName.substring(3);  // value of studentName remain as "Martin"
studentName = studentName.substring(3);
```



Equality of two strings

- Use equals() instead of "==" to compare for equality of two strings
  - Example: "bcd" == "abcd".substring(1) ➔ false



- Syntax:
  - boolean equals(String anotherStr)
  - Example: "bcd".equals("abcd".substring(1)) ➔ true

Compare two strings

- The method int compareTo(String str) compares two strings lexicographically
- Example: str1.compareTo(str2) returns
  - integer > 0 if str1 > str2
  - integer = 0 if str1 = str2
  - integer < 0 if str1 < str2

String Comparison Examples

str1.compareTo(str2)

str1	str2	return value	reason
“AAAA”	“ABCD”	<0	‘A’ <‘B’
“aaaa”	“AAAA”	>0	‘a’ > ‘A’
“127”	“409”	<0	‘1’ < ‘4’
“abc12”	“abc12”	=0	equal string
“abc”	“abcde”	<0	str1 is a sub string of str2
“3”	“12345”	>0	‘3’ > ‘1’

String concatenation

- String concatenation can be done by
  - The addition operator (+)

    Example:

    ```
    String name = “TC Pong”;
    IO.outputln(“My name is “ + name);
    // will print: My name is TC Pong
    ```

  - The method String concat(String str)

    Example:

    ```
    IO.outputln(“My name is ”.concat(name));
    // will print: My name is TC Pong
    ```

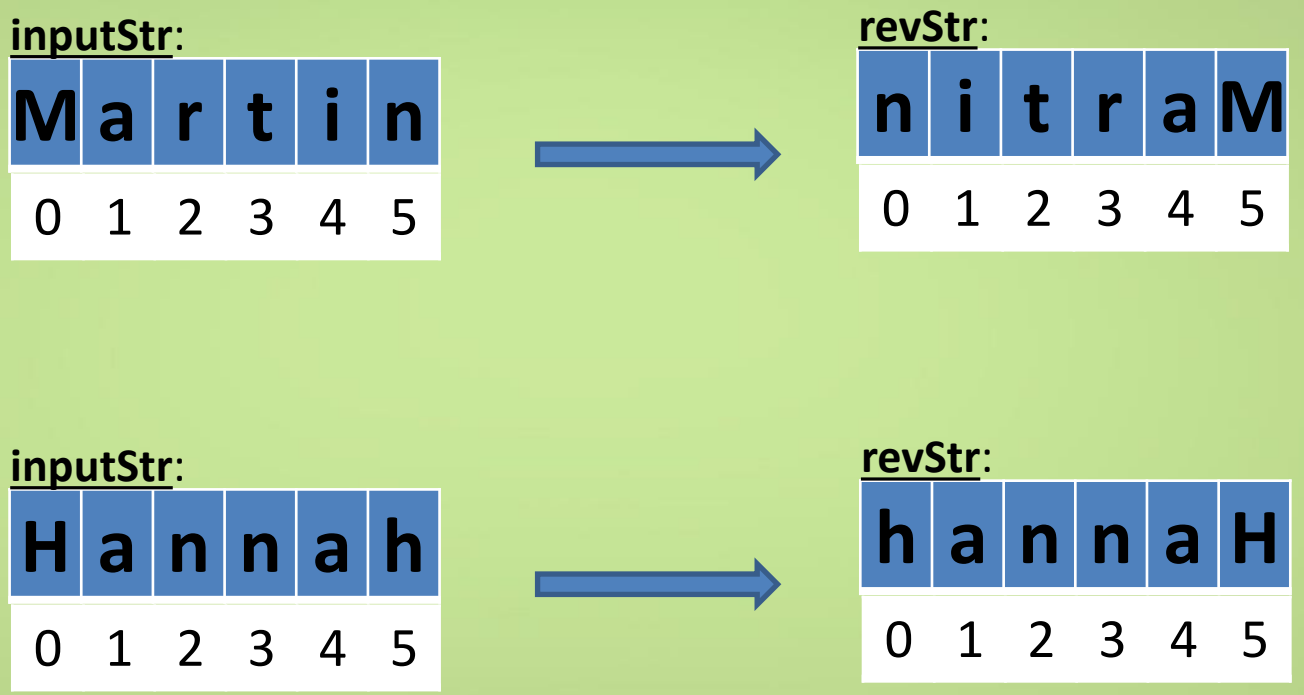
Convert to Lower or Upper case

- The method String toLowerCase( ) converts all characters in the string to lower case
- The method String toUpperCase( ) converts all characters in the string to upper case
- Example:
  - “AbCdE”.toLowerCase( ) returns “abcde”
  - “AbCdE”.toUpperCase( ) returns “ABCDE”

More String Methods

- http://docs.oracle.com/javase/6/docs/api/java/lang/String.html

Reverse a string



Reverse a string

```
public String reverseString(String inputStr) {
   String revStr = "";

   for (int i=0; i < inputStr.length( ); i++) {
         revStr = inputStr.charAt(i) + revStr;
   }

   return revStr;
}
```

i = 0

inputStr:

M	a	r	t	i	n
0	1	2	3	4	5

inputStr.charAt(0):

M

revStr:

M

Reverse a string

```
public String reverseString(String inputStr) {
   String revStr = "";

   for (int i=0; i < inputStr.length( ); i++) {
         revStr = inputStr.charAt(i) + revStr;
   }

   return revStr;
}
```

i = 1

inputStr:

M	a	r	t	i	n
0	1	2	3	4	5

inputStr.charAt(1):

a

+

revStr:

a	M

Reverse a string

```
public String reverseString(String inputStr) {
   String revStr = "";

   for (int i=0; i < inputStr.length( ); i++) {
         revStr = inputStr.charAt(i) + revStr;
   }

   return revStr;
}
```

i = 2

inputStr:

M	a	r	t	i	n
0	1	2	3	4	5

inputStr.charAt(2):

r

+

revStr:

r	a	M

Reverse a string

```
public String reverseString(String inputStr) {
   String revStr = "";

   for (int i=0; i < inputStr.length( ); i++) {
         revStr = inputStr.charAt(i) + revStr;
   }

   return revStr;
}
```

i = 3

inputStr:

M	a	r	t	i	n
0	1	2	3	4	5

inputStr.charAt(3):

t

+

revStr:

t	r	a	M

Reverse a string

```
public String reverseString(String inputStr) {
   String revStr = "";

   for (int i=0; i < inputStr.length( ); i++) {
         revStr = inputStr.charAt(i) + revStr;
   }

   return revStr;
}
```

i = 4

inputStr:

M	a	r	t	i	n
0	1	2	3	4	5

inputStr.charAt(4):

i

+

revStr:

i	t	r	a	M

Reverse a string

```
public String reverseString(String inputStr) {
   String revStr = "";

   for (int i=0; i < inputStr.length( ); i++) {
         revStr = inputStr.charAt(i) + revStr;
   }

   return revStr;
}
```

i = 5

inputStr:

M	a	r	t	i	n
0	1	2	3	4	5

inputStr.charAt(5):

n

+

revStr:

n	i	t	r	a	M

Reverse a string

```
public String reverseString(String inputStr) {
   String revStr = "";

   for (int i=0; i < inputStr.length( ); i++) {
         revStr = inputStr.charAt(i) +  revStr;
   }

   return revStr;
}
```

i = 6

inputStr:

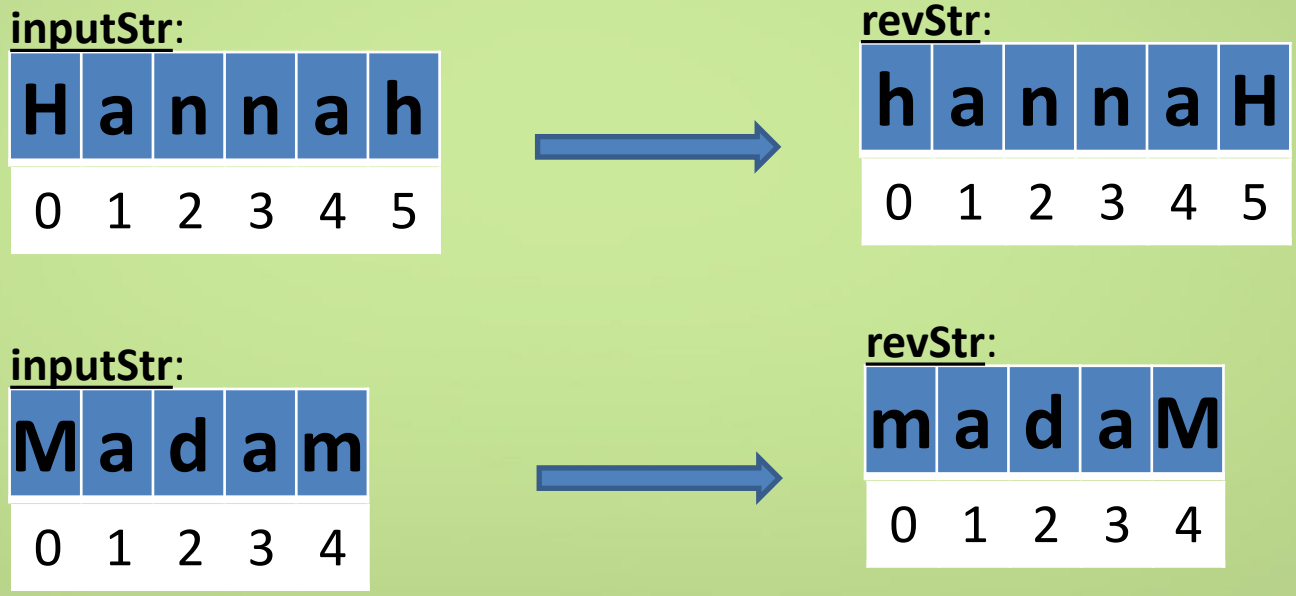
M	a	r	t	i	n
0	1	2	3	4	5

revStr:

n	i	t	r	a	M

Example: Palindrome

- A palindrome is a word or sentence which reads the same in both directions.



Palindrome

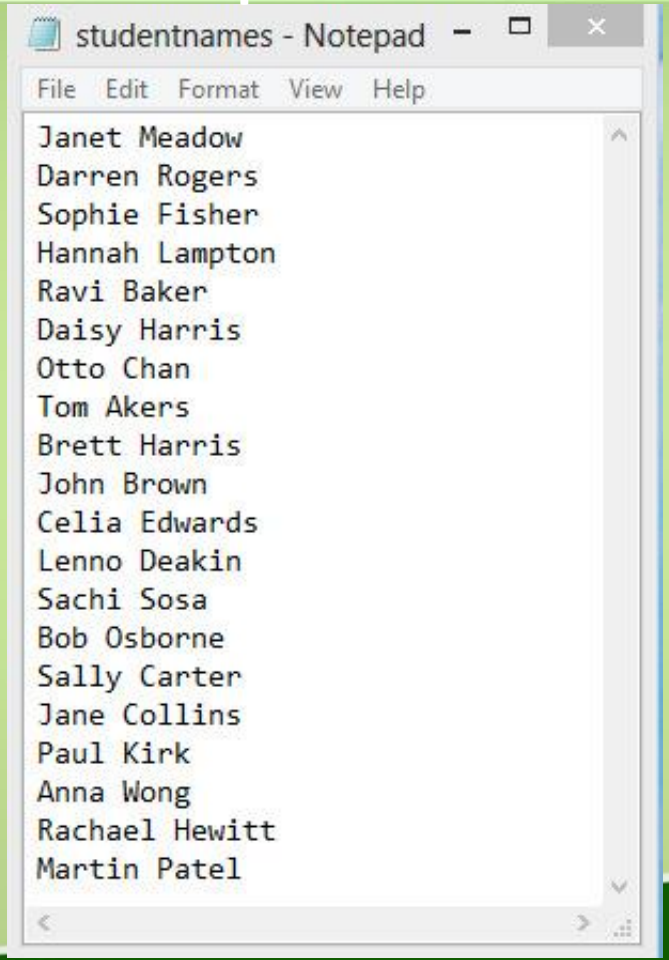
- Write a method isPalindrome which returns true if the input parameter is a palindrome and returns false otherwise.

```
public boolean isPalindrome(String str) {

      // hint:  use the method reverseString
}
```

File I/O

A sample text file



File I/O

- Three steps of file I/O
  1. open/create a file
  2. read/write/do something with the file
  3. close the file
- We are using File/PrintWriter/Scanner classes for this course
  - These are Java's pre-defined classes, we need to import the corresponding libraries

Importing libraries

To make use of the pre-defined classes of File I/O in Java, add these lines to the beginning of code:

```
// The first few lines…
import java.io.File;
import java.io.PrintWriter;
import java.util.Scanner;

// The remaining parts…
```

Exception handling

- Exception handling is necessary when performing file I/O
  - E.g. an incorrect filename, error in opening a file
- Use try-catch block for exception handling

```
try {
    //  try  block
} catch (ExceptionName e) {
    //  catch block
}
```

- To workaround with exception handling, you need to add "throws IOException" next to the method header that reads/writes a file

```
public void doSomeFileIO()  throws IOException {
   //  some File I/O code…
}
```

File Input: Using Scanner

- Steps to use a Scanner
  1. Create a File object using a given filename
  2. Create a Scanner object using the File object
  3. Read data from file and process the data
  4. Close the file by using close() method
- A Scanner breaks its input into tokens using a delimiter pattern (whitespace by default).
- The resulting tokens may then be converted into values of different types (including primitive types and strings).

Useful methods in Scanner class

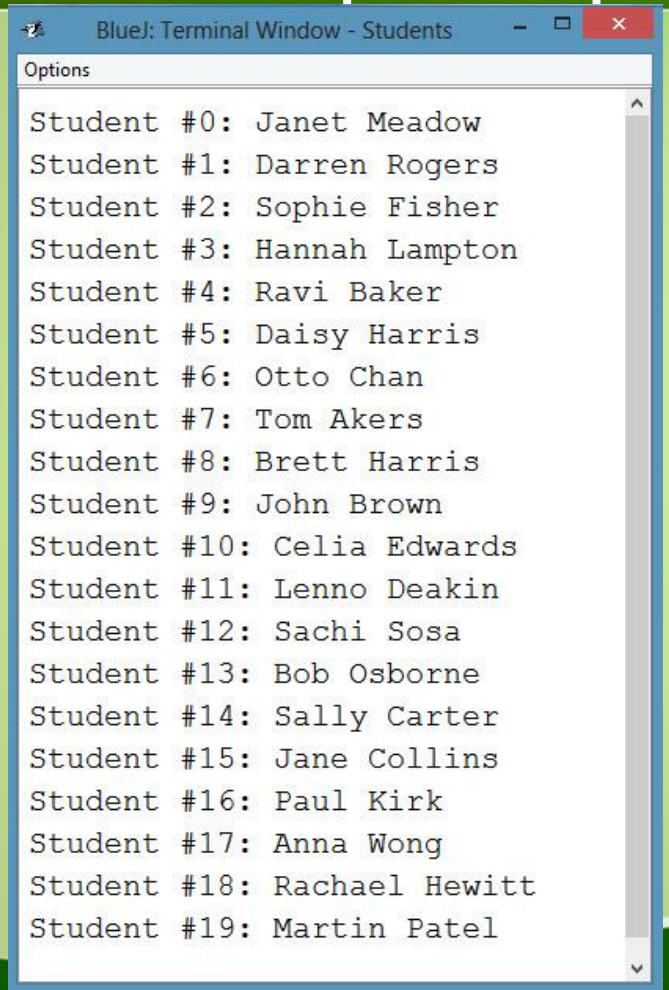
- Read content from a file:
  - nextLine()
    - Returns the next line of string
  - next()
    - Returns the next string (only the first will be returned if the line contains many words)
  - nextInt(), nextDouble()
    - Reads the next string and try to convert it to an integer (int) / a floating point number (double)
- Check the availability of the content from a file:
  - hasNextLine()
    - Returns true if the next line exists
  - hasNext()
    - Returns true if the next string exists
  - hasNextInt(), hasNextDouble()
    - Returns true if the next int/double exists

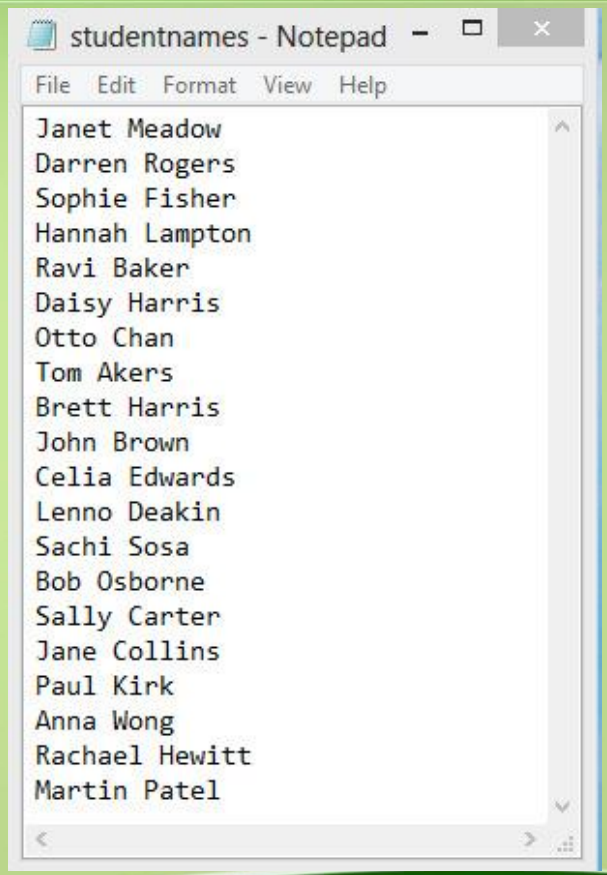
http://download.oracle.com/javase/6/docs/api/java/util/Scanner.html

Example: Read the names of students

```
public void readStudentNamesFromFile() throws Exception {
  // 1. Create a File and Scanner objects
  File inputFile = new File("studentnames.txt");
  Scanner input = new Scanner(inputFile);
  // 2. read the content using a loop
  for (int i=0;  input.hasNextLine(); i++) {
       String inputStudentName = input.nextLine();
       IO.outputln("Student #" + i + ": " + inputStudentName);
  }
  // 3. close the file and print the result
  input.close();
}
```

Example: Input and output





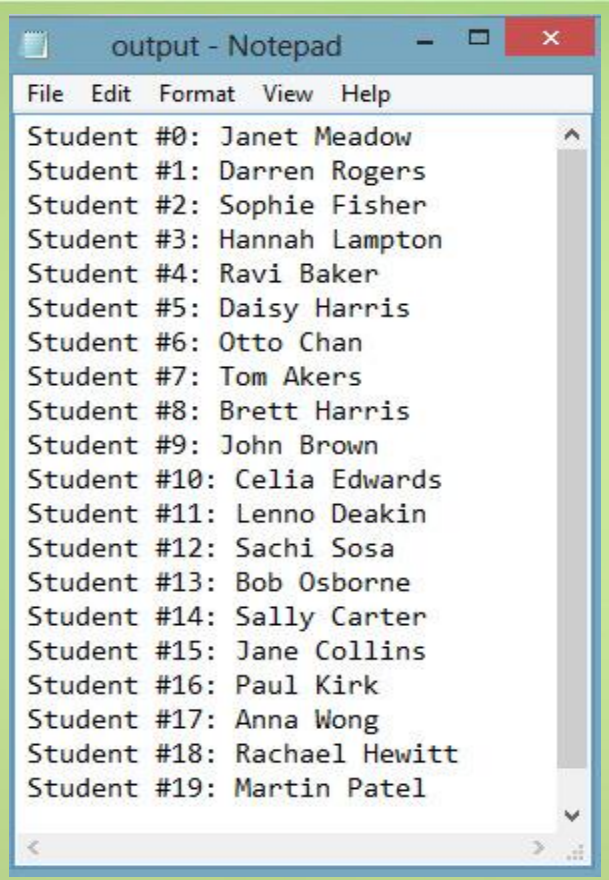
File Output: Using PrintWriter

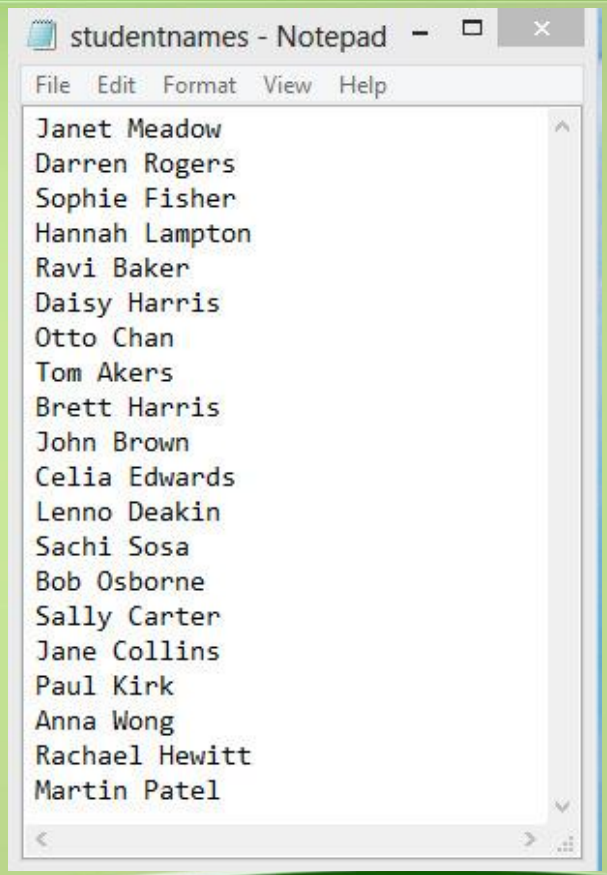
- Steps to use a PrintWriter
  1. Create a File object by a given filename
  2. Create a PrintWriter object using the File object
  3. Output data to the file
     - print( ) or println( )
  4. Invoke close() method to close the file

Example: Output results to a file

```
public void readWriteStudentNames () throws IOException {
  // 1.1 Create a File and Scanner objects
  File inputFile = new File(“studentnames.txt");
  Scanner input = new Scanner(inputFile);
  // 1.1 Create a File and PrintWriter objects
  File outputFile = new File("output.txt");
  PrintWriter writer = new PrintWriter(outputFile);
  // 2. read and outputthe content using a loop
  for (int i=0;  input.hasNextLine(); i++) {
       String inputStudentName = input.nextLine();
    writer.println (“Student #” + i + “: “ + inputStudentName);
  }
  // 3. close the file and print the result
  input.close();
  writer.close();
}
```

Example: Input and output files





Input from OCR

Optical Character Recognition (OCR)

- Automatic conversion of printed or handwritten text into machine readable text.
- Applications:
  - Postal address from envolops
  - Bank transactions
  - Reading for the visually impaired
  - Handwriting input, e.g. signature verification
  - Vehicles license plate numbers
  - others

Example: Read text from OCR

```
import comp102x.IO;

public class inputFromOCR
{
   private static Loader loader = new Loader(); // for loading OCR libraries

   public void readFromOCR() {
      // Input from image by performing Optical Character Recognition(OCR)
      String text = IO.inputTextImage();

      IO.outputln(text);

   }
}
```

https://www.edx.org/course/hkustx/hkustx-comp102x-introduction-computing-1690#.U9rAT_mSzQM

edX HOW IT WORKS COURSES SCHOOLS & PARTNERS dashboard

Introduction to Computing with Java

Designed to equip students with the fundamental elements of programming and data abstraction using Java.

Java.

Estimated effort: 3 - 5 hours/week

Access Courseware

About this Course

Do you wish to become a better problem solver?

This course aims to provide you with a good understanding of basic Java

Java IO Tutorial.htm | lec18 System-in-read.pdf | IowaEngineer2013N....pdf | IMG_20140729_213223.jpg | IMG_20140729_073720.jpg | Show all downloads...

6:17 AM 8/1/2014


HKUST 8


HKUST 18

Example: Read the names of students

```
public void readStudentNamesFromFile() throws IOException {
  // 1. Create a File and Scanner objects
  File inputFile = new File("studentnames.txt");
  Scanner input = new Scanner(inputFile);
  // 2. read the content using a loop
  for (int i=0;  input.hasNextLine(); i++) {
       String inputStudentName = input.nextLine();
       IO.outputln("Student #" + i + ": " + inputStudentName);
  }
  // 3. close the file and print the result
  input.close();
}
```

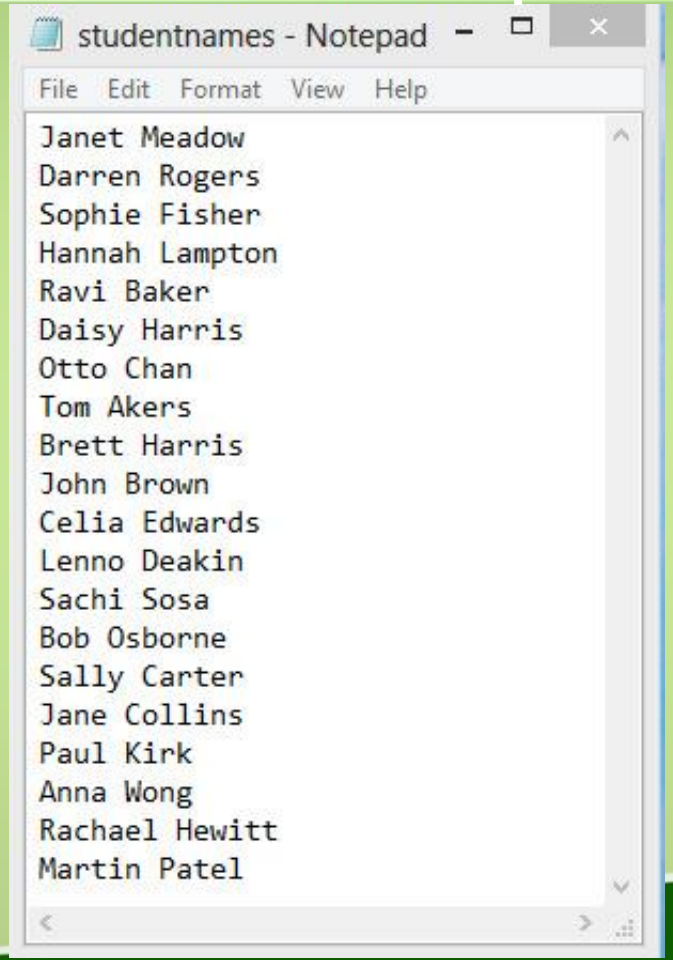
Example: Read names from Console

```
public void readStudentNamesFromFile() throws IOException {
  // 1. Create a File and Scanner objects
  File inputFile = new File(“studentnames.txt");
  Scanner input = new Scanner(inputFile);
  // 2. read the content using a loop
  for (int i=0;  input.hasNextLine(); i++) {
       String inputStudentName = input.nextLine();
       IO.outputln(“Student #” + i + “: “ + inputStudentName);
  }
  // 3. close the file and print the result
  input.close();
}
```

Example: Read names from Console

```
public void readNamesFromConsole() throws IOException {
    // 1. Create Scanner objects for standard input
    Scanner input = new Scanner(System.in);
    int nStudents = 0;
    // 2. read the content using a loop
    while (true) {
            String inputName = input.nextLine();
            if (inputName.equals("")) break;
            IO.outputln("Student #" + nStudents + ": " + inputName);
            nStudents++;
        }
}
```

Scanner breaks input into tokens

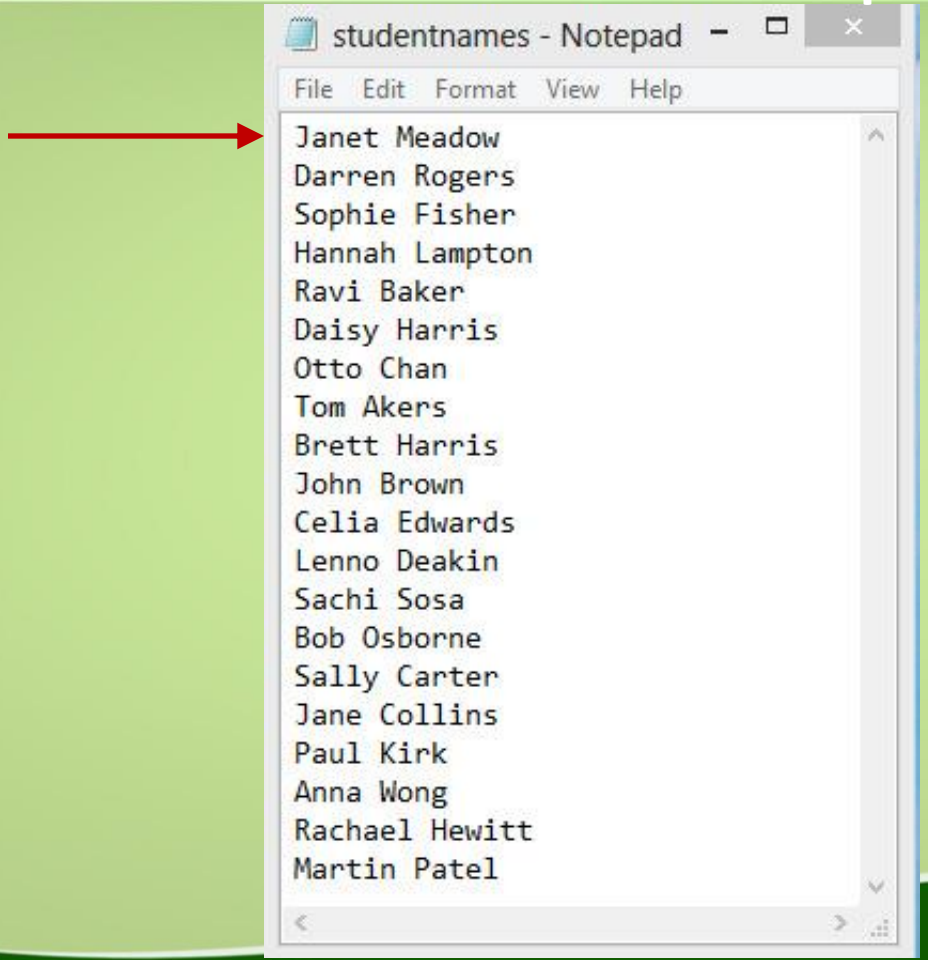


Example: Read names into an array

```
public class Student2DArray {
   private static final int maxN = 40;  // Assuming there are no more than 40 names
   private String[ ][ ] studentNames = new String[maxN][2]; // For 1st and last names
   private int nStudents = 0;     // Number of students

   public void readStudentNamesToArray() throws IOException {
      // 1. Create a File and Scanner objects
      File inputFile = new File("studentnames.txt");
      Scanner input = new Scanner(inputFile);
      Scanner line;     // A scanner object for each line of input
      // 2. read the content and then store in an array using a loop
      for (int i=0;  input.hasNextLine(); i++) {
         String inputStudentName = input.nextLine();
         line = new Scanner(inputStudentName);
         if (i >= maxN) break;
         studentNames[i][0] = line.next();
         studentNames[i][1] = line.next();
         nStudents++;
      }
      // 3. close the file and print the result
    input.close();
   }
```

Scanner can break input into tokens



Example: Read names into an array

```
public class StudentNameArray {
   private static final int maxN = 40;  // Assuming there are no more than 40 names
   private String[ ][ ] studentNames = new String[maxN][2]; // For 1st and last names
   private int nStudents = 0;     // Number of students

   public void readStudentNamesToArray() throws IOException {
      // 1. Create a File and Scanner objects
      File inputFile = new File("studentnames.txt");
      Scanner input = new Scanner(inputFile);
      Scanner line;     // A scanner object for each line of input
      // 2. read the content and then store in an array using a loop
      for (int i=0;  input.hasNextLine(); i++) {
         String inputStudentName = input.nextLine();
         line = new Scanner(inputStudentName);
         if (i >= maxN) break;
         studentNames[i][0] = line.next();
         studentNames[i][1] = line.next();
         nStudents++;
      }
      // 3. close the file and print the result
      input.close();
   }
```

"Janet Meadow"

Example: Output the Array to a File

```
public void outputNameArrayToFile( ) throws IOException {
    // 1.1 Create a File and PrintWriter objects
    File outputFile = new File("output2.txt");
    PrintWriter writer = new PrintWriter(outputFile);
    // 2. read the content and then store in an array using a loop
    for (int i=0;  i < nStudents; i++) {
        // The last name is output before the first name
        writer.println(studentNames[i][1] + ", " + studentNames[i][0]);
    }
     // 3. close the file and print the result
    writer.close();
  }
}
```