SE102 Abstract Data Type and Problem Solving
Software Engineering

Friday November 11, 11

Lecture 1

Agenda

- Course Syllabus
- Cheating Agreement
- Program Submission
- Pre-Test
- Basic I/O in Java

Course Syllabus

- Prerequisite
  - SE101 Computers and Programming
- Instructors
  - Section 701, 9:30 – 11:00am, Tuesday, Friday
  - Room 213 CAMT building
  - Pree Thienburanathum
  - preenet@gmail.com, pree.t@cmu.ac.th
  - Office Room 417

  - Section 702, 9:30 – 11:00am, Tuesday, Friday
  - Room RB 5402
  - Parinya Suwansrikham
  - aj.parinya@hotmail.com
  - Office Room 417

  - TA (to be announce)

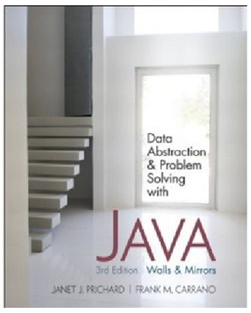
Course Syllabus

Course Objectives

- After completing this course, students will be able to:
  - Explain basic knowledge in abstract data types, information hiding and modularity
  - Apply fundamental data structures to represent data for problem solving
  - Analyze and design the abstract data types for program design
  - Implement techniques of computer-based problem solving using ADTs

Course Syllabus

- Course Texts
- 1.Data Abstraction and Problem Solving with Java, 3rd edition, 2011, Frank Carrano, Janet Prichard


Data Abstraction & Problem Solving with
JAVA
3rd Edition Walls & Mirrors
JANET J. PRICHARD | FRANK M. CARRANO

Course Syllabus

- **Course Website**
  Please register and check the course announcement all homework assignments will be post at the following website.
  http://cmuonline.cmu.ac.th/

Course Syllabus

- **Course Requirements**
  – Lectures in class
  – Quizzes and Programming assignments
  – Reading assignments
  – Paper-based exams

Course Syllabus

Grading System

- The semester grade is computed as group:

– 4 x Programming Assignments	***30%***
– Attendance	***5%***
– 4 x Quizzes	***5%***
– Midterm Examination	***30%***
– Final Examination	***30%***
– **Total**	**100%**

Cheating Agreement Form

- Please complete the Agreement Form and turn them to me.

Program Submission

- We will have 4 X Programming Assignments in this course.
- Late submission will not be accepted!
- Put the source code (.java) and **README** file into a CD
- Print out the source code (.java)
- Print out the **README** file
- Put everything in to the brown envelop.
- Seal and write your name, student id and your partner one.

Pre-Test

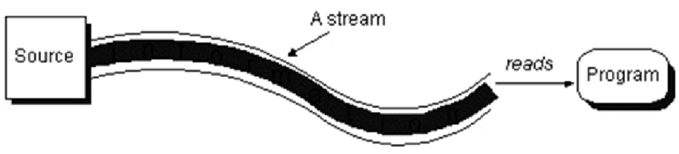
- This is to get us and idea the level of your programming skill after completed the SE101.
- You have 30 minutes to complete the pre-test.

- 5 Minutes Break

Reading & Writing Data

- Data can come from many Sources & go to many Destinations
  - Memory
  - Disk
  - Network
- Whatever the Source or Destination, a Stream has to be opened to Read/Write Data

Reading & Writing Data



Reading

Open a Stream

While more Information

Read

Close the Stream

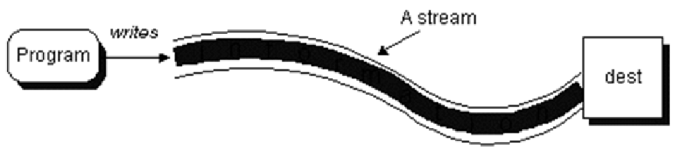
Writing

Open a Stream

While more Information

Write

Close the Stream

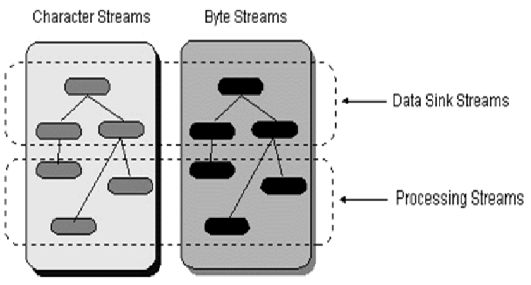


Reading & Writing Data

- java.io Package includes these Stream Classes
  - *Character Streams* are used for 16-bit Characters – Uses *Reader* & *Writer* Classes
  - *Byte Streams* are used for 8-bit Bytes – Uses *InputStream* & *OutputStream* Classes Used for Image, Sound Data etc.

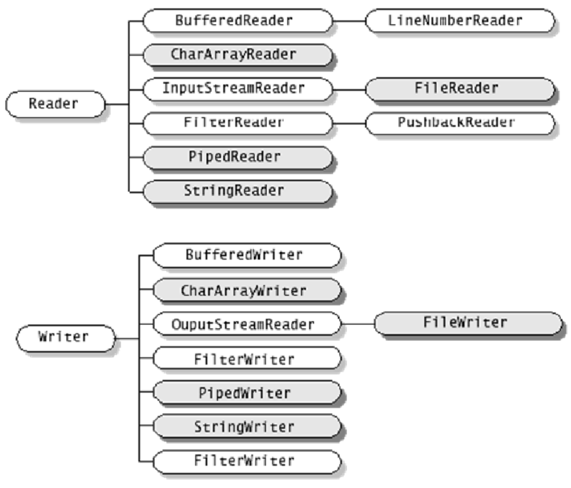
Reading & Writing Data

- Data Sinks
  - Files
  - Memory
  - Pipes
- Processing
  - Buffering
  - Filtering



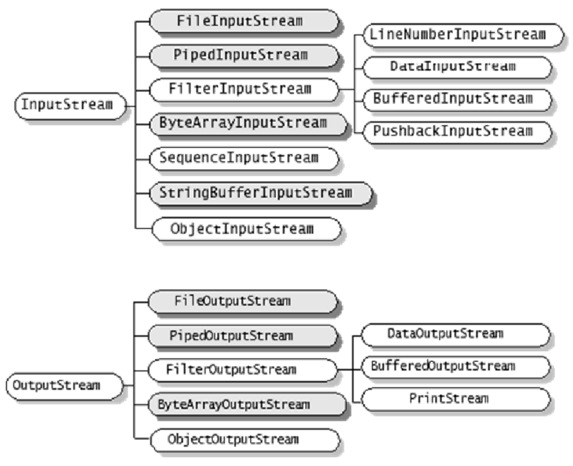
Character Streams

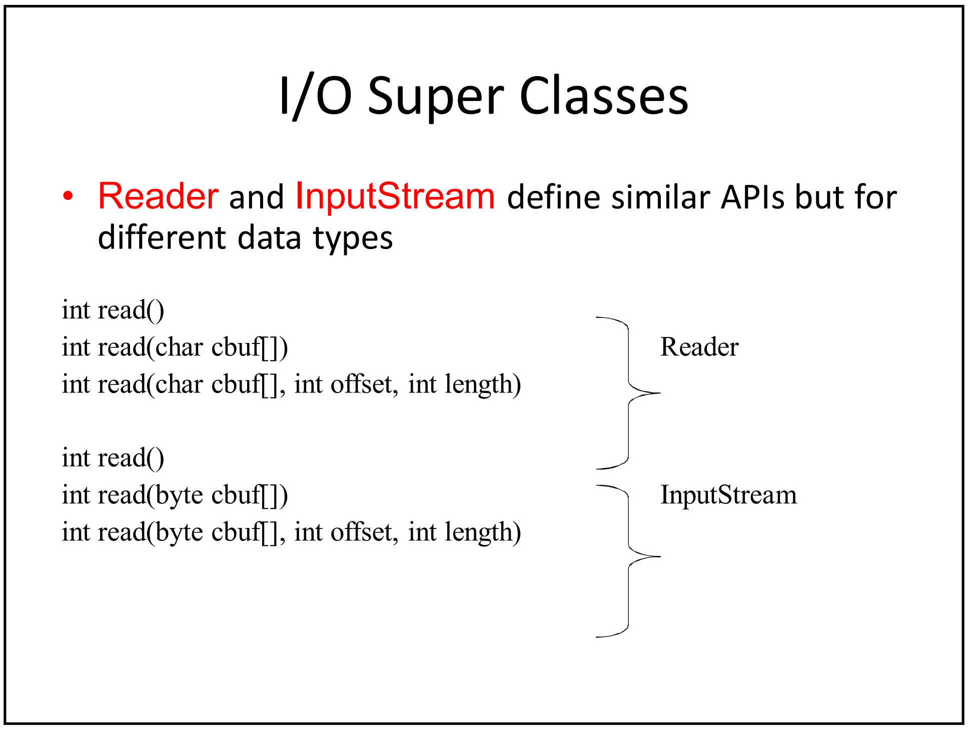
- Reader and Writer are abstract super classes for character streams (16-bit data)
- Sub classes provide specialized behavior



Byte Streams

- InputStream and OutoutStream are abstract super classes for byte streams (8-bit data)
- Sub classes provide specialized behavior



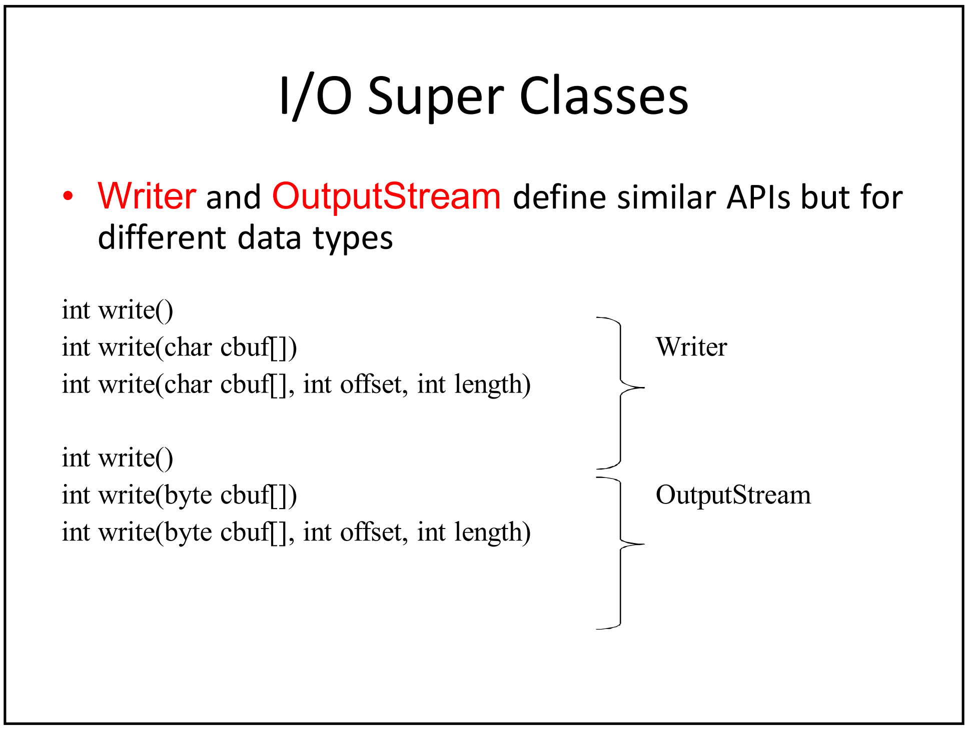


I/O Super Classes

- Reader and InputStream define similar APIs but for different data types

```
int read()
int read(char cbuf[])                                   Reader
int read(char cbuf[], int offset, int length)

int read()
int read(byte cbuf[])                                   InputStream
int read(byte cbuf[], int offset, int length)
```



I/O Super Classes

- Writer and OutputStream define similar APIs but for different data types

```
int write()
int write(char cbuf[])                                  Writer
int write(char cbuf[], int offset, int length)

int write()
int write(byte cbuf[])                                  OutputStream
int write(byte cbuf[], int offset, int length)
```

Type of I/O	Streams	Description
Memory	CharArrayReader CharArrayWriter ByteArrayInputStream ByteArrayOutputStream	Use these streams to read from and write to memory. You create these streams on an existing array and then use the read and write methods to read from or write to the array.
	StringReader StringWriter StringBufferInputStream	Use StringReader to read characters from a String in memory. Use StringWriter to write to a String. StringWriter collects the characters written to it in a StringBuffer, which can then be converted to a String. StringBufferInputStream is similar to StringReader, except that it reads bytes from a StringBuffer.
Pipe	PipedReader PipedWriter PipedInputStream PipedOutputStream	Implement the input and output components of a pipe. Pipes are used to channel the output from one thread into the input of another.
File	FileReader FileWriter FileInputStream FileOutputStream	Collectively called file streams, these streams are used to read from or write to a file on the native file system.
Object Serializati-on	*N/A* ObjectInputStream ObjectOutputStream	Used to serialize objects.

Stream wrapping

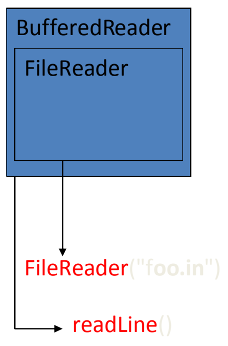
- BufferedReader class can be used for efficient reading of characters, arrays and lines

  BufferedReader in = new BufferedReader(new FileReader("foo.in"));

- BufferedWriter and PrintWriter classes can be used for efficient writing of characters, arrays and lines and other data types

BufferedWriter out = new BufferedWriter(newFileWriter("foo.out"));

PrintWriter out= new PrintWriter(new BufferedWriter(new FileWriter("foo.out")));



Getting User Input in Command Line

- Read as reading from the standard input device which is treated as an input stream represented by System.in

```
BufferedReader input= new
      BufferedReader(newInputStreamReader(System.in));
System.out.println("Enter the name :"  );
String name =input.readLine();
```