**SFDV3006 Concurrent Programming**
**Assignment 1: Concurrent Programming using Java threads**

## Problem 1: [3 marks] Implement a version of Java *grep* using pipes and filters

Write a version of *grep* utility in Java. You program must take arguments which are the text to search and the file to search. The program would be used as follows:

*$ java JGrep import Buffer.java*

The *JGrep* class must create and start separate threads for the filter and the generator to search for word in the given file, for example as shown above search for word *import* in the file *Buffer.java*

The word to search and the file to search are supplied as *command line arguments*, which can be accessed from the *main()* method of a class.

Output of the JGrep: Your solution must produce an output as shown below:
*$ java JGrep import Buffer.java*
*1 : Buffer.java import java.util.*;*

Where *1* is the line number where the word was found in the file. *Buffer.java* is the file where it was found and *import.java.util.*;* is the line which has the import word.

Java classes you will need to solve Problem 1:
1. *BufferedReader*, *FileReader* from *java.io* package
2. *StringBuffer* from *java.util* package
3. *Thread* class and its methods such as *join()*
4. *String* class and its methods such as *indexOf()*

You must also know how to access arguments from the *main()* method.

## Problem 2: [3 marks] Mulithreaded file finder

Write a multi threaded version of the UNIX *find* command which is used to find file in directories. The program would be used as follows:

*$ java JFind Buffer.java C:\javaprogs C:\programs*

In the above command the first argument *Buffer.java* is the file to search and other two arguments are directories where the file (*Buffer.java*) is to be searched.

For every directory start a new thread and pass it the file to search and the name of the directory.

In this problem, every thread prints its own result - an example of which is shown below:

*$ java JFind Buffer.java C:\javaprogs C:\programs*
*Buffer.java found in C:\javaprogs*
*Buffer.java not found C:\programs*
It is very common that directories contain subdirectories. When a thread finds that there is a directory inside another directory, it creates a separate thread to search the subdirectory.

Java classes you will need to solve Problem 2:

1. You must use the *File* class in *java.io* package. This class represents both files and directories. From this class you will need to use *isFile()*, *isDirectory()* and *list()* method which return arrays of files/directories.

You must also know how to access arguments from the *main()* method. You must also know how to loop through arrays to search and how to use break statement.

**Problem 3 : Dining Savages problem [Condition synchronization : 4 marks]**

Consider the following problem:

"The problem of the dining savages (an allusion to the classic dining philosophers) is based on the arguably tasteless analogy of a number of members of a primitive culture, hereinafter called the "savages", sharing a meal from a single pot. The primary abstractions are the savages themselves, a cook, and the pot. The pot contains a certain number of servings of savage sustenance (the nature of which will be left to your imagination). Each of the savages can freely remove a serving from the pot so long as the pot is not empty. So before taking a serving, a savage must check to make sure the pot is not empty. In the case in which the pot is empty, the savage must wake up the cook to refill the pot, after which the feast continue. The savages, then, can eat only when the pot is not empty, and the cook can fill the pot only when the pot is empty".

Identify the threads and the monitor and implement your solution in Java to synchronize the savages and the cook.

---

**Issue date:** 18 April 2011

**Due date:** 02 May 2011 (see also section on late submission below)

**Weightage:** 10%.
To get full marks for a question - your classes and programs must behave correctly in all situations and you must implement the correct number of classes, threads, monitors and programs as required

**Working in groups:** Students can work in groups of up to a maximum of 3.

**Copying:** Copying whether in part or full, from other students or other sources will strictly not be tolerated leading to a loss of 100% (for this assignment) for all groups involved.

**Late submission:** 20% of weightage for this assignment is deducted per day after due date.

**Submission guidelines:** Do a demo in the labs and submit the zip file for each of the questions by email as attachments to ***assignments@sfdv3006.tk***