**CA4 \_ hints**

For reading the text-file

*Agror  
Dwarf  
Warrior  
14  
12, 15, 19, 9, 14, 7*

You will probably use a scanner. The first 4 lines are simple, use ***scan.nextLine().***

***After that the 5th line are read:***

***String abilities=scan.nextLine(); // and then what ??***

Apparently each number is separated by ',' and some 'white-space':

If you use a second scanner:

***Scanner lineScanner=new Scanner(abilities);***

You can't use ***lineScanner.nextInt(),*** because the ',' is a decimal separator(at least in Denmark)!

The normal separator for 'tokens' is 'white-space', so when each token is evaluated the comma will be there. An easy fix would be to change the file-format to:

*Agror  
Dwarf  
Warrior  
14  
12 15 19 9 14 7*

***BUT THAT IS A VIOLATION OF THE SPECIFICATION!!!***

The solution is to change the delimiter definition:

***Scanner lineScan = new Scanner(abilities); lineScan.useDelimiter( ",\\s\*" );***

***Now you can use lineScan.nextInt();***

The string ",\\s\*" defines the ***pattern***, that should be used to separate tokens -   
in this case a ',' followed by zero ore more whitespace-characters.

\s is defined as any character from the following set :[0x20 \t \n 0x0B \f \r]

0x20 = 'space'  
\t = 'tab'  
\n = 'new line'  
0x0B = 'vertical tab'  
\f = 'form feed'  
\r = ' carriage-return'

The asterix '\*' specifies 'zero or more'

So the pattern is **",\s\*"** ,but '\' is an escape-character in java so we must write is as ",\\s\*"

**Loading the GamerCollection from file**

Lets assume that bt default data is stored in either a text-file:  
"gamers.txt" or serialized to an object-file:"gamers.dat".

When the program starts it should try to load the 'gamers' from the "gamers.dat" file, but if the file isn't there, it should try the "gamers.txt" file. If we cant find the textfile, just create an empty collection.

**private** GamerCollection gamers;

**private** **void** initializeGamers()

{

File f = **new** File("gamers.dat");

**if** (f.exists())

{

**try**

{

gamers= ***<deserialise(f)>*** //you code this part

} **catch** (ClassNotFoundException e)

{

e.printStackTrace();

} **catch** (FileNotFoundException e)

{

e.printStackTrace();

} **catch** (IOException e)

{

e.printStackTrace();

}

} **else**

{

gamers = **new** GamerCollection();

addFromtextFile("gamers.txt");

}

}

After reading the text-file, serialize the collection to the gamers.dat file.

Remember that **each time you change something in either the GamerCollection-class or in the Gamer-class, you have to delete the object-file (gamers.dat)**

**Using a file-chooser (swing-component):**

**If you put a 'load from text-file' – Button in the gui, you could use a file-chooser (in order to avoid mis-spelling's):**

**public** **void** actionPerformed(java.awt.event.ActionEvent e)

{

JFileChooser chooser = **new** JFileChooser(System

.*getProperty*("user.dir")); //project-directory

FileNameExtensionFilter filter = **new** FileNameExtensionFilter(

"TextFiles", "txt"); //only display \*.txt files

chooser.setFileFilter(filter);

**int** returnVal = chooser.showOpenDialog(**null**);

**if** (returnVal == JFileChooser.*APPROVE\_OPTION*)

{

controller.addFromtextFile(chooser

.getSelectedFile().getName());

}

}