(20102340) Jorja Holland

20102340@mail.wit.ie

Abstract

This portfolio contains practical 1 through 7 and a Linux Final Project

Operating Systems portfolio

Jorja Holland 20102340

Contents

[***SECTION 1 1***](#_Toc152273093)

[***Practical 1 1***](#_Toc152273094)

[***Practical 2 5***](#_Toc152273095)

[***Practical 3 8***](#_Toc152273096)

[***Practical 4 13***](#_Toc152273097)

[***Practical 5 16***](#_Toc152273098)

[***Section 2 20***](#_Toc152273099)

[***Practical 6 20***](#_Toc152273100)

[***Section 3 22***](#_Toc152273101)

[***Practical 7 22***](#_Toc152273102)

**START OF PORTFOLIO**

**START OF SECTION 1**

# ***SECTION 1***

START OF PRATICAL 1

## ***Practical 1***

*1.Exploring the use of commands with options and arguments Taking a selection of Windows CLI commands from those given below, use the online help to examine the various options and arguments, and try them out. You can look up others at:* [*http://www.microsoft.com/resources/documentation/windows/xp/all/proddocs/enus/ntcmds.mspx?mfr=true*](http://www.microsoft.com/resources/documentation/windows/xp/all/proddocs/enus/ntcmds.mspx?mfr=true)

*You're required carefully to write two A4 pages (Times 12 point or equivalent size) detailing your experiments with different options for between six and ten different commands.*

*To get the online help for a command,*

*type command /? 6. e.g. dir /?*

*prompt*

*mkdir*

*color*

*title*

*tree*

*type*

*ver*

*print*

*xcopy*

*Type help at the windows command line prompt to see some more instructions.*

*ANSWER(S)*

The 7 commands I have chosen to experiment with were dir, prompt, mkdir, title, tree, xcopy and print.

**Dir**

The command dir lists the files and subdirectories in a directory. This command displays the disk’s volume label, model number and folder list and files. It provides details such as the name of the files and the date it was last accessed. The command also displays the overall quantity of listed files and folders, their overall size, and the remaining free disk space in bytes.

I used this command when conducting my labs to gain a better understanding of the command. I learned how it shows my entire directory and I then further learned how to navigate through my directory using the command cd. I carried out some of the examples on arranging the files in alphabetical order which was very efficient, and I learned how to only look for certain files such as text files which was very helpful.

**Prompt**

**My understanding of the command:**

The prompt command alters the cmd.exe command prompt, which allows you to display any text the way you like, including the current directory’s name the time and the date. This command restores the command prompt to its default state, which is the current drive letter and directory followed by the greater than symbol (>), if it is used without any parameters.

**My experience with the command:**

I had to conduct additional study on the command prompt, so I tested out the examples on the Microsoft Learn website which the link is provided on the question, and it helped me gain some further understanding as to what the function of the prompt command is.

**Mkdir**

**My understanding of the command:**

The mkdir command allows you to establish a directory or subfolder is a task. With the help of a single mkdir command and command extensions which are by default enabled you can build between folders in each path.

**My experience with the command:**

My own understanding of this command is that you can create directories and sub-directories decided to add to my knowledge of this command by trying out the examples given on the Microsoft Learn website linked in the question above.

I learned how to create a directory tree with the titles students\teachers\colleges within the root directory but with the command extensions disabled. This was the result of experimenting with this example.

**Title**

**My understanding of the command:**

The title of an MS-DOS window can be changed with the title command.

**My experience with the command:**

I learned how to change the title of the Command Prompt window to Updating Files while the batch file executes the copy command and then change it back to Command Prompt my using the Microsoft Learn examples.

**Tree**

**My understanding of the command:**

Provides a graphic depiction of the disk in a drive or the directory structure of a path. The structure that this program shows depends on the options you specify at the command line. If you don't give a drive or path, this command displays the tree structure beginning with the current directory of the current drive.

**My experience with the command:**

I experimented with the tree command as I am new to this command and I furthered my knowledge by trying the examples in Microsoft Learn which showed me how to display all the names of the subdirectories on the disk in my current drive, display one screen at a time all the files in all the directories on drive C and to print a list of all the directories on drive C to a file.

A screenshot of a computer

Description automatically generated

**Ver**

**My understanding of the command:**

Displays windows version.

**My experience with the command:**

I had to type the command help into my command prompt to understand the command ver. From my understanding of the command, it displays the Windows version of the command prompt.

**colour**

**My understanding of the command:**

changes the background and foreground colours in the Command Prompt window for the current session. When used without any parameters, colour returns the Command Prompt window's foreground and background colours.

**My experience with the command:**

By reading through the examples and testing them out in my command prompt I thoroughly enjoyed playing around with the command and I was pleased to learn more about the colours I could change the background.

*2.What’s the purpose of the first line - @ECHO OFF? Remove it and see the effect.*

*ANSWER(S)*

The line @ECHO OFF is used to stop characters entered at the keyboard from echoing or showing up on the screen.

**USING @ECHO OFF**

A screenshot of a computer

Description automatically generated

**NOT USING @ECHO OFF**

A computer screen shot of a computer program

Description automatically generated

END OF PRACTICAL 1

START OF PRACTICAL 2

## ***Practical 2***

1. *Test the above using a directory that you created last week.*

*ANSWER(S)*

A screen shot of a computer

Description automatically generated



1. *Simply record what happens when you run this script. 2.3 Find out how to run scripts if they’re in a directory other than the working directory. 2.4 Study the following script and see if you can figure out what it’ll do. Now type it into a file called game.ps1 (You can use copy/paste in places to reduce the labour.) Run it and see if your predictions are true.*

*ANSWER(S)*

A screenshot of a computer program

Description automatically generated





*5. Create another file called beverage.txt identical to drink.txt. (Hint: Use what you learned in practical 1 to do this.)*

*6. Issue this command again PS C:\Users\User1\green> Get-ChildItem | select-string coffee and see what happens.*

*7. Create a file called fruit.txt with the list apple orange banana in it.*

*8. Issue this command again PS C:\Users\User1\green> Get-ChildItem | select-string coffee and see what happens.*

*ANSWER(S) TO Q5,Q6,Q7,Q8 BELOW*

A computer screen shot of a black screen

Description automatically generated

*16. Things to Do.*

*Concluding Things to Do Watch the video at the first of the two links below and then (using the video or other sources) answer the following questions.*

*9. Find out and explain how to get help about any cmdlet.*

*ANSWER(S)*

The command Get-Help is very useful in PowerShell. The short description provided in PowerShell after you type the command is as follows “Displays help about Windows Powershell cmdlets and concepts”. A cmdlet typically refers to a short script that is designed to carry out a single, focused task, such as copying files or rearranging directories. When you type the command Get-Help it displays numerous headings but the one I am interested in is the heading which details how to get help specifically with cmdlets. The command shown to get help with cmdlets is as follows: Get-Help <cmdlet-name>.

A screenshot of a computer program

Description automatically generated

*10. Find out and explain what F7 does in PowerShell.*

*ANSWER(S)*

F7 displays the command history.

*11. What is the purposes of (a) the –whatif switch and (b) the –confirm.*

*ANSWER(S)*

Whatif switch

the whatif switch commands basically show what would happen if you executed a command without executing the command.

- confirm switch

The confirm switch tells the command which it is applied to pause processing the command before it executes. The command then requests your permission to execute each command making you acknowledge what each command does.

*12. Write a note to explain how you can use tab to complete a command as soon as it’s unambiguous.*

*ANSWER(S)*

By simply pressing tab on your keyboard and the shell will autofill the command for you. This is useful in commands that are similar bar maybe a few letters or numbers which in turn improves efficiency when operating PowerShell.

END OF PRACTICAL 2

START OF PRACTICAL 3

## ***Practical 3***

1. *Simply record what happens when you run this script. What difference does it make if you leave out the text "Please type your name" from the first line of the script?*

*ANSWER(S)*

A screenshot of a computer

Description automatically generated

A screen shot of a computer program

Description automatically generated

A screenshot of a computer

Description automatically generated

A black background with white text

Description automatically generated

1. *Find out how to run scripts if they’re in a directory other than the working directory.*

*ANSWER(S)*



*3. (a) Find out how to do the same thing that the code above does except that it’ll only accept integers (such as 67).*

*(b) Once you’re found the answer, find out what happens if you type a real number (such as 67.4 or 67.8)*

*ANSWER(S)*

A black text on a white background

Description automatically generated

A screen shot of a computer

Description automatically generated

A screen shot of a computer

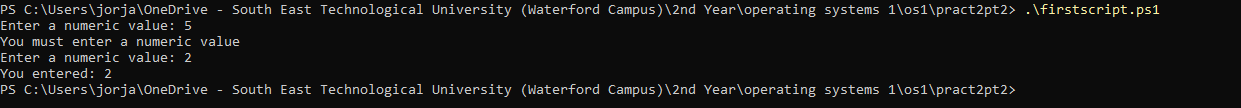
Description automatically generated

1. *Alter the above program (in brown) to require the user to enter specifically integer values between 1 and 4 inclusive.*

*ANSWER(S)*

A screenshot of a computer program

Description automatically generated



1. *What does the above script do?*

*ANSWER(S)*

A white background with black text

Description automatically generated

i= 1 at the start and while i is less than or equal to 20, if i is not equal to 13 and the same for 17 print out the number.

1. *We don’t need the brackets round ($i -ne 13) and ($i -ne 17) in the code above. Why do we not need them? (Hint: The answer is the same as for Java). Do you think that it’s a good idea to put them in even if they aren’t necessary? Explain your answer.*

*ANSWER(S)*

I believe that it is not necessary to have brackets in, but it is a good idea to make the code easier to read.

A computer screen shot of a black screen

Description automatically generated

1. *We can replace $i = $i + 1 with something shorter, in the above two scripts. What do you think it is? Try it and see.*

*ANSWER(S)*

A white background with black text

Description automatically generatedA white background with red text

Description automatically generated

1. *Oops, I've put semicolons at the end of each of the lines, in the do...until loop above. I suppose it's because of my experience in writing programs in other languages that sometimes I put semicolons at the end of a line of PowerShell script, even when they're entirely unnecessary in PowerShell. Does PowerShell forgive me for doing this? Find out and write your conclusion.*

*ANSWER(S)*

It doesn’t care for semicolons as can be seen in the example below where I took out the semicolons.

A close-up of a math equation

Description automatically generatedA black text with white text

Description automatically generated with medium confidence

A screen shot of a computer

Description automatically generated

1. *Alter this program to deal with grade categories (for example >70 is a distinction mark etc) in an examination and allow the user to enter a grade.*

*ANSWER(S)*

A white background with black text

Description automatically generatedA computer code with white text

Description automatically generated

1. *Explain what the above example does. Modify it to show fields other than the Name field.*

*ANSWER(S)*

The first example shown when implemented shows the names of the files in that directory. I changed it to show the length of the directory.

A close-up of a math equation

Description automatically generated

A screen shot of a computer code

Description automatically generated

END OF PRACTICAL 3

START OF PRACTICAL 4

## ***Practical 4***

1. *Create a directory tree structure (similar to the structure outlined in Practical 1)*

*ANSWER(S)*

A computer screen shot of a line of words

Description automatically generated

1. *Find out what happens when you type time at the command line in (a) UNIX and (b) Windows CLI.*

*ANSWER(S)*

1. A black text on a white background

   Description automatically generated
2. A black background with white text

   Description automatically generated
3. *As an extension to the script NU.SH, you used the shell script in conjunction with the file redirection operator (>) to redirect the output to a file called howmany;*

*ANSWER(S)*

i.e. $ ./nu.sh > howmany.

*Run the command again but this time with the >> operator. Check the contents of the file and explain the output.*

*ANSWER(S)*

The >> operator changes a present file or creates a new file if that file name hasn’t been used already in a directory

*4. Show the finished shell script code (including comments) for example 2 exercise. To show date and time, the number of files and directories and your current working directory.*

*5.What’s the output of the stats.sh program, and what’s the significance of the echo statement on the line without any succeeding text.*

*ANSWER(S) TO Q4 AND Q5*

With the echo statement it shows the writing after it but it also puts a space between each statement such as echo the current time and date date echo echo the number of files in my system are ls -l | wc -l

The first screenshot shows the script code with the echo statement.

A black and white text

Description automatically generated

This screenshot shows the script without the echo statement.

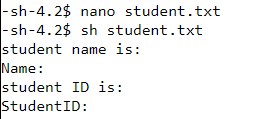
A screenshot of a computer

Description automatically generated

*6. Write a shell script (STUDENTS.SH) to ask the User to enter two fields. Their NAME and STUDENT-ID. The script should append the data to a text file called STUDENTS.TXT.*

*Put the relevant commands into a script and run the script, remembering to use chmod to make your script executable.*

*ANSWER(S)*

A black text on a white background

Description automatically generated

END OF PRATICAL 4

**START OF PRACTICAL 5**

## ***Practical 5***

1. *Find out what is listed by $ ls ~*

*ANSWER(S)*

A black and white striped background

Description automatically generated

1. *Find out what is listed by $ ls ~/..*

*ANSWER(S)*

A screenshot of a computer

Description automatically generated

*3.Find out what is listed by $ cd~/*

*ANSWER(S)*

A white background with black text

Description automatically generated

*4.Using the above method create another file called list 2 containing the following fruit: orange, plum, mango, grapefruit. List the contents of list 2.*

*ANSWER(S)*

A screenshot of a computer program

Description automatically generated

1. *Find out what happens if you do*

*cat >>mondaylist*

*rabbit*

*hare*

*^d*

*And mondaylist doesn’t exist yet.*

*ANSWER(S)*

A white background with black text

Description automatically generated

*6.Find out what happens if you do*

*cat >>mondaylist*

*rabbit*

*hare*

*^d*

*And mondaylist does exist yet.*

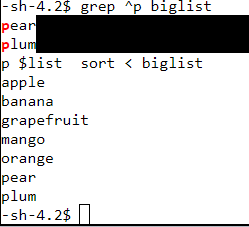
*ANSWER(S)*

A screenshot of a computer program

Description automatically generated

1. *Using pipes, display all lines of list1 and list2 containing the letter ‘p’ and sort the result.*

*ANSWER(S)*



1. *Using pipes and filters search all files ending in \*.txt in your linuxstuff directory that contain the names of ALL other users in your class currently logged in. Output the results in ascending order to Standard Output.*

*ANSWER(S)*

A white background with black text

Description automatically generated

*Exercises*

*Explain what the following do:*

*ANSWER(S)*

1. *ls linuxstuff/backups*

lists the contents of your backup directories.

1. *ls ~/linuxstuff*

it is used to specify paths starting at your home directory.

1. *ls ~*

it is used to specify paths starting at a directory.

1. *ls ~/..*

goes back one path.

1. *cat > list1 … ^D*

creates a list.

1. *cat >> list1 … ^D*

edit a list

1. *cat list1*

shows list

1. *cat list1 list2 biglist*

combines both lists

1. *sort … ^D*

sort alphabetically or numerically sorts a list.

1. *sort < biglist*

sort alphabetically or numerically sorts the biglist.

1. *sort < biglist > slist*

sort alphabetically or numerically sorts a biglist and makes a new list for the sorted list.

1. *ls list\**

This will list all files in the current directory starting with list

1. *ls \*list*

This will list all files in the current directory ending with list

1. *ls ?list*

picks a random list (wildcard).

1. *who*

To see who is on the system at the same time as you are.

1. *who > names.txt sort < names.txt who | sort*

sorts text files with the names of people who is on the system.

1. *who | wc –l*

find out how many users are logged on.

1. *grep 'orange' list2*

print out each line in the file which contained the string orange.

1. *grep '^grape' list2*

print out each line in the file which started with the string grape.

1. *grep 'grape$' list2*

all lines ending in grape.

*Write a note to explain the difference between ~ and / in describing directories.*

When using the / it specifies the paths starting at your home directory and when using ~ it is used to specify paths starting at a directory. The command ~ doesn’t specify the directory whereas the / command the users directly puts in the directory.

END OF SECTION

START OF SECTION

# ***SECTION 2***

## ***Practical 6***

Source code for menu.sh, loginmenu.sh, register.txt and register.sh.

Menu.sh

#!/bin/sh

echo WELCOME, PLEASE CHOOSE FROM ONE OF THE

echo FOLLOWING OPTIONS

echo

echo

echo 1.REGISTER YOUR DETAILS

echo

echo

echo

echo 2.LOGIN TO YOUR ACCOUNT

echo

read -p "Enter one of the options" option

case $option in

1)

sh ./register.sh

;;

\*)

echo go to login menu

;;

esac

**loginmenu.sh**

#!/bin/sh

echo WELCOME TO LOGIN MENU

echo

echo 1.LOGIN

echo

echo

read -p "Enter the option to login" option

case $option in

1)

sh ./login.sh

;;

\*)

echo maybe you dont have an account go to main menu

;;

esac

register.sh

#Account name:

#DOB:

#Username:

#Password:

#Email Address:

read -p "Enter Account Name" Accountname

read -p "Enter DOB" DOB

read -p "Enter Username" Username

read -p "Enter Password" Password

read -p "Enter Email Address" EmailAddress

echo $name , $dob , $username , $password , $email >> register.txt

Register.txt

Jorja123 , 22/09/03 , jorjaHolland , 12345 , jorjaholland89@gmail.com

Ava321 , 07/01/02 , avaaaa , arklow , avaneary2020@outlook.com

Georgie , 25/12/19 , Santa , NorthPole , santa25@elves.com

, , ,

**finding specific attributes about the registration**

**EXAMPLE**

-sh-4.2$ cut register.txt -d: -f 1,4|grep Jorja

**Jorja**123 , 22/09/03 , jorjaHolland , 12345 , jorjaholland89@gmail.com

-sh-4.2$ cut register.txt -d: -f 1,4|grep Jorja123

**Jorja123** , 22/09/03 , jorjaHolland , 12345 , jorjaholland89@gmail.com

-sh-4.2$ cut register.txt -d: -f 1,4|grep Ava

**Ava**321 , 07/01/02 , avaaaa , arklow , avaneary2020@outlook.com

-sh-4.2$ cut register.txt -d: -f 1,4|grep G

**G**eorgie , 25/12/19 , Santa , NorthPole , santa25@elves.com

**Comments**

**Proving that users exist**

To test that the scripts work and the .txt file worked I used the grep, pipe and cut command to test that there were users to be found in the register.txt file. I tested each individual account to see that by using the commands specified the user existed.

**END OF SECTION**

**START OF SECTION**

# ***SECTION 3***

**START OF PRATICAL 7**

## ***Practical 7***

*Thread Program 1.1*

*Things to try part 1.1:*

*1. Create an Eclipse Java Project to use this program and run the software 10 times, each time copying and pasting your results into a word processor file. Label this set of 10 results clearly with the heading Threading Program 1.1*

*Do you see much variation on your results from one run to the next? Explain what conclusions you come to.*

*2. If possible, repeat the exercise above with computers that are single core, dual*

*core and quad core. Is there a difference in how these machines perform?*

*ANSWER(S)*

**First time running**

bbbbbbbbbbbbbbbbbbbbbbbbbbbbbbbbbbbbbbbbbbbbbbbbbbbbbbbbbbbbbbbbbbbbbbbbbbbbbbbbbbbbbbbbbbbbbbbbbbbbaaaaaaaaaaaaaaaaaaaaaaaaaaaaaaaaaaaaaaaaaaaaaaaaaaaaaaaaaaaaaaaaaaaaaaaaaaaaaaaaaaaaaaaaaaaaaaaaaaaa 1 2 3 4 5 6 7 8 9 10 11 12 13 14 15 16 17 18 19 20 21 22 23 24 25 26 27 28 29 30 31 32 33 34 35 36 37 38 39 40 41 42 43 44 45 46 47 48 49 50 51 52 53 54 55 56 57 58 59 60 61 62 63 64 65 66 67 68 69 70 71 72 73 74 75 76 77 78 79 80 81 82 83 84 85 86 87 88 89 90 91 92 93 94 95 96 97 98 99 100

Process finished with exit code 0

**Second time running**

aaaaaaaaaaaaaaaaaaaaaaaaaaaaaaaaaaaaaaaaaaaaaaaaaaaaaaaaaaaaaaaaaaaaaaaaaaaaaaaaaaaaaaaaaaaaaaaaaaaabbbbbbbbbbbbbbbbbbbbbbbbbbbbbbbbbbbbbbbbbbbbbbbbbbbbbbbbbbbbbbbbbbbbbbbbbbbbbbbbbbbbbbbbbbbbbbbbbbbb 1 2 3 4 5 6 7 8 9 10 11 12 13 14 15 16 17 18 19 20 21 22 23 24 25 26 27 28 29 30 31 32 33 34 35 36 37 38 39 40 41 42 43 44 45 46 47 48 49 50 51 52 53 54 55 56 57 58 59 60 61 62 63 64 65 66 67 68 69 70 71 72 73 74 75 76 77 78 79 80 81 82 83 84 85 86 87 88 89 90 91 92 93 94 95 96 97 98 99 100

Process finished with exit code 0

**Third time running**

aaaaaaaaaaaaaaaaaaaaaaaaaaaaaaaaaaaaaaaaaaaaaaaaaaaaaaaaaaaaaaaaaaaaaaaaaaaaaaaaaaaaaaaaaaaaaaaaaaaabbbbbbbbbbbbbbbbbbbbbbbbbbbbbbbbbbbbbbbbbbbbbbbbbbbbbbbbbbbbbbbbbbbbbbbbbbbbbbbbbbbbbbbbbbbbbbbbbbbb 1 2 3 4 5 6 7 8 9 10 11 12 13 14 15 16 17 18 19 20 21 22 23 24 25 26 27 28 29 30 31 32 33 34 35 36 37 38 39 40 41 42 43 44 45 46 47 48 49 50 51 52 53 54 55 56 57 58 59 60 61 62 63 64 65 66 67 68 69 70 71 72 73 74 75 76 77 78 79 80 81 82 83 84 85 86 87 88 89 90 91 92 93 94 95 96 97 98 99 100

Process finished with exit code 0

**Fourth time running**

aaabbbbbbbbbbbbbbbbbbbbbbbbbbbbbbbbbbbbbbbbbbbbaaaaaaaaaaaaaaaaaaaaaaaaaaaaaaaaaaaaaaaaaaaaaaaaaaaaaaaaaaaaaaaaaaaaaaaaaaaaaaaaaaaaaaaaaaaaaaaaa 1 2 3 4 5 6 7 8 9 10 11 12 13 14 15 16 17 18 19 20 21 22 23 24 25 26 27 28 29 30 31 32 33 34 35 36 37 38 39 40 41 42 43 44 45 46 47 48 49 50 51 52 53 54 55 56 57 58 59 60 61 62 63 64 65 66 67 68 69 70 71 72 73 74 75 76 77 78 79 80 81 82 83 84 85 86 87 88 89 90 91 92 93 94 95 96 97 98 99 100bbbbbbbbbbbbbbbbbbbbbbbbbbbbbbbbbbbbbbbbbbbbbbbbbbbbbbbb

Process finished with exit code 0

**Fifth time running**

aaaaaaaaaaaaaaaaaaaaaaaaaaaaaaaaaaaaaaaaaaaaaaaaaaaaaaaaaaaaaaaaaaaaaaaaaaaaaaaaaaabbbbbbbbbbbbbbbbbbbbbbbbbbbbbbbbbbbbbbbbbbbbbbbbbbbbbbbbbbbbbbbbbbbbbbbbbbbbbbbbbbbbbbbbbbbbbbbbbbbbaaaaaaaaaaaaaaaaa 1 2 3 4 5 6 7 8 9 10 11 12 13 14 15 16 17 18 19 20 21 22 23 24 25 26 27 28 29 30 31 32 33 34 35 36 37 38 39 40 41 42 43 44 45 46 47 48 49 50 51 52 53 54 55 56 57 58 59 60 61 62 63 64 65 66 67 68 69 70 71 72 73 74 75 76 77 78 79 80 81 82 83 84 85 86 87 88 89 90 91 92 93 94 95 96 97 98 99 100

Process finished with exit code 0

**Sixth time running**

abbbbbbbbbbbbbbbbbbbbbbbbbbbbbbbbbbbbbbbbbbbbbbbbbbbbbbbbbbbbbbbbbbbbbbbbbbbbbbbbbbbbbbbbbbbbbbbbbbbbaaaa 1 2 3 4 5 6 7 8 9 10 11 12 13 14 15 16 17 18 19 20 21 22 23 24 25 26 27 28 29 30 31 32 33 34 35 36 37 38 39 40 41 42 43 44 45 46 47 48 49 50 51 52 53 54 55 56 57 58 59 60 61 62 63 64 65 66 67 68 69 70 71 72 73 74 75 76 77 78 79 80 81 82 83 84 85 86 87 88 89 90 91 92 93 94 95 96 97 98 99 100aaaaaaaaaaaaaaaaaaaaaaaaaaaaaaaaaaaaaaaaaaaaaaaaaaaaaaaaaaaaaaaaaaaaaaaaaaaaaaaaaaaaaaaaaaaaaaa

Process finished with exit code 0

**Seventh time running**

abbbbbbbbbbbbbbbbbbbbbbbbbbbbbbbbbbbbbbbbbbbbbbbbbbbbbbbbbbbbbbbbbbbbbbbbbbbbbbbbbbbbbbbbbbbbbbbbbbbbaaaa 1 2 3 4 5 6 7 8 9 10 11 12 13 14 15 16 17 18 19 20 21 22 23 24 25 26 27 28 29 30 31 32 33 34 35 36 37 38 39 40 41 42 43 44 45 46 47 48 49 50 51 52 53 54 55 56 57 58 59 60 61 62 63 64 65 66 67 68 69 70 71 72 73 74 75 76 77 78 79 80 81 82 83 84 85 86 87 88 89 90 91 92 93 94 95 96 97 98 99 100aaaaaaaaaaaaaaaaaaaaaaaaaaaaaaaaaaaaaaaaaaaaaaaaaaaaaaaaaaaaaaaaaaaaaaaaaaaaaaaaaaaaaaaaaaaaaaa

Process finished with exit code 0

**Eighth time running**

aaaaaaabbbbaaaaaaaaaaaaaaaaaaaaaaaaaaaaaaaaaaaaaaaaaaaaaaaaaaaaaaaaaaaaaaaaaaaaaaaaaaaaaaaaaaaaaaaaaaaaabbbbbbbbbbbbbbbbbbbbbbbbbbbbbbbbbbbbbbbbbbbbbbbbbbbbbbbbbbbbbbbbbbbbbbbbbbbbbbbbbbbbbbbbbbbbbbbb 1 2 3 4 5 6 7 8 9 10 11 12 13 14 15 16 17 18 19 20 21 22 23 24 25 26 27 28 29 30 31 32 33 34 35 36 37 38 39 40 41 42 43 44 45 46 47 48 49 50 51 52 53 54 55 56 57 58 59 60 61 62 63 64 65 66 67 68 69 70 71 72 73 74 75 76 77 78 79 80 81 82 83 84 85 86 87 88 89 90 91 92 93 94 95 96 97 98 99 100

Process finished with exit code 0

**Nineth time running**

aaaaaaaaaaaaaaaaaaaaaaaaaaaaaaaaaaaaaaaaaaaaaaaaaaaaaaaaaaaaaaaaaaaaaaaaaaaaaaaaaaaaaaaaaaaaaaaaaaaabbbbbbbbbbbbbbbbbbbbbbbbbbbbbbbbbbbbbbbbbbbbbbbbbbbbbbbbbbbbbbbbbbbbbbbbbbbbbbbbbbbbbbbbbbbbbbbbbbbb 1 2 3 4 5 6 7 8 9 10 11 12 13 14 15 16 17 18 19 20 21 22 23 24 25 26 27 28 29 30 31 32 33 34 35 36 37 38 39 40 41 42 43 44 45 46 47 48 49 50 51 52 53 54 55 56 57 58 59 60 61 62 63 64 65 66 67 68 69 70 71 72 73 74 75 76 77 78 79 80 81 82 83 84 85 86 87 88 89 90 91 92 93 94 95 96 97 98 99 100

Process finished with exit code 0

**Tenth time running**

bbbbbbbbbbbbbbbbbbbbbbbbbbbbbbbbbbbbbbbbbbbbbaabbbbbbbbbbbbbbbbbbbbbbbbbbbbbbbbbbbbbbbbbbbbbbbbbbbbbbbaaaaaaaaaaaaaaaaaaaaaaaaaaaaaaaaaaaaaaaaaaaaaaaaaaaaaaaaaaaaaaaaaaaaaaaaaaaa 1 2 3 4 5 6 7 8 9 10 11 12 13 14 15 16 17 18 19 20 21 22 23 24 25 26 27 28 29 30 31 32 33 34 35 36 37 38 39 40 41 42 43 44 45 46 47 48 49 50 51 52 53 54 55 56 57 58 59 60 61 62 63 64 65 66 67 68 69 70 71 72 73 74 75 76 77 78 79 80 81 82 83 84aaaaaaaaaaaaaaaaaaaaaa 85 86 87 88 89 90 91 92 93 94 95 96 97 98 99 100

Process finished with exit code 0

*3. Replace the lines:*

*thread1.start();*

*thread2.start();*

*thread3.start();*

*with*

*thread1.run();*

*thread2.run();*

*thread3.run();*

*Does your software still run?*

*If it does, what difference do you see?*

*ANSWER(S)*

**When you change it from thread.start to thread.run**

aaaaaaaaaaaaaaaaaaaaaaaaaaaaaaaaaaaaaaaaaaaaaaaaaaaaaaaaaaaaaaaaaaaaaaaaaaaaaaaaaaaaaaaaaaaaaaaaaaaabbbbbbbbbbbbbbbbbbbbbbbbbbbbbbbbbbbbbbbbbbbbbbbbbbbbbbbbbbbbbbbbbbbbbbbbbbbbbbbbbbbbbbbbbbbbbbbbbbbb 1 2 3 4 5 6 7 8 9 10 11 12 13 14 15 16 17 18 19 20 21 22 23 24 25 26 27 28 29 30 31 32 33 34 35 36 37 38 39 40 41 42 43 44 45 46 47 48 49 50 51 52 53 54 55 56 57 58 59 60 61 62 63 64 65 66 67 68 69 70 71 72 73 74 75 76 77 78 79 80 81 82 83 84 85 86 87 88 89 90 91 92 93 94 95 96 97 98 99 100

Process finished with exit code 0

The differences I noticed is that the program still runs but there is no variation it just prints 100 times b 100 times and goes from 0 - 100. Whereas in the other program, it has thread.start there was a variation which is visible in the running of it 10 times you can see the pattern of the a's and b's changing positions.

*Thread Program 1.2*

*Things to try part 1.2:*

*1. Create a Java Eclipse Project to use this program and run the software 10  
 times. Each time copy and paste your results into a word processor file. Label  
 this set of 10 results clearly with the heading Threading Program 1.2*

*Do you see much variation in your results from one run to the next? Explain  
 what conclusions you come to.*

*ANSWER(S)*

**First time running**

aaaaaaaaaaaaaaaaaaaaaaaaaaaaaaaaaaaaaaaaaaaaaaaaaaaaaaaaaaaaaaaaaaaaaaaaaaaaaaaaaaaaaaaaaaaaaaaaaaaabbbbbbbbbbbbbbbbbbbbbbbbbbbbbbbbbbbbbbbbbbbbbbbbbbbbbbbbbbbbbbbbbbbbbbbbbbbbbbbbbbbbbbbbbbbbbbbbbbbbcccccccccccccccccccccccccccccccccccccccc 1 2 3 4 5 6 7 8 9 10 11 12 13 14 15 16 17 18 19 20 21 22 23 24 25 26 27 28 29 30 31 32 33 34 35 36 37 38 39 40 41 42 43 44 45 46 47 48 49 50 51 52 53 54 55 56 57 58 59 60 61 62 63 64 65 66 67 68 69 70 71 72 73 74 75 76 77 78 79 80 81 82 83 84 85 86 87 88 89 90 91 92 93 94 95 96 97 98 99 100

Process finished with exit code 0

**Second time running**

aaaaaaaaaaaaaaaaaaaaaaaaaaaaaaaaaaaaaaaaaaaaaaaaaaaaaaaaaaaaaaaaaaaaaaaaaaaaaaaaaaaaaaaaaaaaaaaaaaaabbbbbbbbbbbbbbbbbbbbbbbbbbbbbbbbbbbbbbbbbbbbbbbbbbbbbbbbbbbbbbbbbbbbbbbbbbbbbbbbbbbbbbbbbbbbbbbbbbbb 1 2 3 4 5 6 7 8 9 10 11 12 13 14 15 16 17 18 19 20 21 22 23 24 25 26 27 28 29 30 31 32 33 34 35 36 37 38 39 40 41 42 43 44 45 46 47 48 49 50cccccccccccccccccccccccccccccccccccccccc 51 52 53 54 55 56 57 58 59 60 61 62 63 64 65 66 67 68 69 70 71 72 73 74 75 76 77 78 79 80 81 82 83 84 85 86 87 88 89 90 91 92 93 94 95 96 97 98 99 100

Process finished with exit code 0

**Third time running**

abbbbbbbaaaaaaaaaaaaaaabbbbbbbbbbbbbbbbbbbbbbbbbbbbbbbbbbbbbbbbbbbbbbbbbbbbbbbbbbbbbbbbbbbbbbbbbbbbbbbbbbbbbbbbbbbbbaaaaaaaaaaaaaaaaaaaaaaaaaaaaaaaaaaaaaaaaaaaaaaaaaaaaaaaaaaaaaaaaaaaaaaaaaaaaaaaaaaaa 1 2 3 4 5 6 7 8 9 10 11 12 13 14 15 16 17 18 19 20 21 22 23 24 25 26c 27 28 29 30 31 32 33 34 35 36 37 38 39 40 41 42 43 44 45 46 47 48 49 50ccccccccccccccccccccccccccccccccccccccc 51 52 53 54 55 56 57 58 59 60 61 62 63 64 65 66 67 68 69 70 71 72 73 74 75 76 77 78 79 80 81 82 83 84 85 86 87 88 89 90 91 92 93 94 95 96 97 98 99 100

Process finished with exit code 0

**Fourth time running**

aaaaabbbbbbbbbbbbbbbbbbbbbbbbbbbbbbbbbbbbbbbbbbbbbbbbbbbbbbbbbbbbbbbbbbbbbbbbbbbbbbbbbbbbbbbbbbbbbbbbaaaaaaaaaaaaaaaaaaaaaaaaaaaaaaaaaaaaaaaaaaaaaaaaaaaaaaaaaaaaaaaaaaaaaaaaaaaaaaaaaaaaaaaaaaaaaaabbbbcccccc 1 2 3 4 5 6 7 8 9 10 11 12 13 14 15 16 17 18 19 20 21 22 23 24 25 26 27 28 29 30 31 32 33 34 35 36 37 38 39 40 41 42 43 44 45 46 47 48cccccccccccccccccccccccccccccccccc 49 50 51 52 53 54 55 56 57 58 59 60 61 62 63 64 65 66 67 68 69 70 71 72 73 74 75 76 77 78 79 80 81 82 83 84 85 86 87 88 89 90 91 92 93 94 95 96 97 98 99 100

Process finished with exit code 0

**Fifth time running**

aaaaaaaaaaaaaaaaaaaaaaaaaaaaaaaaabbbbbbbbbbbbbbbbbbbbbbbbbbbbbbbbbbbbbbbbbbbbbbbbbbbbbbbbbbbbbbbbbbbbbbbbbbbbbbbbbbbbbbbbbbbbbbbbbbbbaaaaaaaaaaaaaaaaaaaaaaaaaaaaaaaaaaaaaaaaaaaaaaaaaaaaaaaaaaaaaaaaaaacccccccccccccccccccccccccccccccccccccccc 1 2 3 4 5 6 7 8 9 10 11 12 13 14 15 16 17 18 19 20 21 22 23 24 25 26 27 28 29 30 31 32 33 34 35 36 37 38 39 40 41 42 43 44 45 46 47 48 49 50 51 52 53 54 55 56 57 58 59 60 61 62 63 64 65 66 67 68 69 70 71 72 73 74 75 76 77 78 79 80 81 82 83 84 85 86 87 88 89 90 91 92 93 94 95 96 97 98 99 100

Process finished with exit code 0

**Sixth time running**

bbbbbbbbbbbbbbaaaaaaaaaaaaaaaaaaaaaaaaaaaaaaaaaaaaaaaaaaaaaaaabbbbbbbbbbbbbbbbbbbbbbbbbbbbbbbaaaaaaaaaaaaaaaaaaaaaaaaaaaaaaaaaaaabbbbbbbbbbbbbbbbbbbbbbbbbbbbbbbbbbbbbbbbbbbbaaaaaaaabbbbbbbbbbbaaaaaaaacccccccccccccccccccccccccccccccccccccccc 1 2 3 4 5 6 7 8 9 10 11 12 13 14 15 16 17 18 19 20 21 22 23 24 25 26 27 28 29 30 31 32 33 34 35 36 37 38 39 40 41 42 43 44 45 46 47 48 49 50 51 52 53 54 55 56 57 58 59 60 61 62 63 64 65 66 67 68 69 70 71 72 73 74 75 76 77 78 79 80 81 82 83 84 85 86 87 88 89 90 91 92 93 94 95 96 97 98 99 100

Process finished with exit code 0

**Seventh time running**

bbbbbbbbbbbbbbbbbbaaaaaaaaaaaaaaaaaaaaaaaaaaaaaaaaaaaaaaaaaaaaaaaaaaaaaaaaaaaaaaaaaaaaaaaaaaaaaaaaaaaaaaaaaaaaaaaaaaaabbbbbbbbbbbbbbbbbbbbbbbbbbbbbbbbbbbbbbbbbbbbbbbbbbbbbbbbbbbbbbbbbbbbbbbbbbbbbbbbbbc 1 2 3 4 5 6 7 8 9 10 11 12 13 14 15 16 17 18 19 20 21 22 23 24 25 26 27 28 29 30 31 32 33 34 35 36 37 38 39 40 41 42 43 44 45 46 47 48 49 50ccccccccccccccccccccccccccccccccccccccc 51 52 53 54 55 56 57 58 59 60 61 62 63 64 65 66 67 68 69 70 71 72 73 74 75 76 77 78 79 80 81 82 83 84 85 86 87 88 89 90 91 92 93 94 95 96 97 98 99 100

Process finished with exit code 0

**Eighth time running**

aaaaaaaaaaaaaaaaaaaaaaaaaaaaaaaaaaaaaaaaaaaaaaaaaaaaaaaaaaaaaaaaaaaaaaaaaaaaaaaaaaaaaaaaaaaaaaaaaaaabbbbbbbbbbbbbbbbbbbbbbbbbbbbbbbbbbbbbbbbbbbbbbbbbbbbbbbbbbbbbbbbbbbbbbbbbbbbbbbbbbbbbbbbbbbbbbbbbbbb 1 2 3 4 5 6 7 8 9 10 11 12 13 14 15 16 17 18 19 20 21 22 23 24 25 26 27 28 29 30 31 32 33 34 35 36 37 38 39 40 41 42 43 44 45 46 47 48 49 50cccccccccccccccccccccccccccccccccccccccc 51 52 53 54 55 56 57 58 59 60 61 62 63 64 65 66 67 68 69 70 71 72 73 74 75 76 77 78 79 80 81 82 83 84 85 86 87 88 89 90 91 92 93 94 95 96 97 98 99 100

Process finished with exit code 0

**Ninth time running**

aaaabbbbbbbbbbbbbbbbbbbbbbbbbbbbbbbbbbbbbbbbbbbbbbbbbbbbbbbbbbbbbbbbbbbbbbbbbbbbbbbbbbbbbbbbbbbbbbbbbbbbaaaaaaaaaaaaaaaaaaaaaaaaaaaaaaaaaaaaaaaaaaaaaaaaaaaaaaaaaaaaaaaaaaaaaaaaaaaaaaaaaaaaaaaaaaaaaaaacccccccccccccccccccccccccccccccccccccccc 1 2 3 4 5 6 7 8 9 10 11 12 13 14 15 16 17 18 19 20 21 22 23 24 25 26 27 28 29 30 31 32 33 34 35 36 37 38 39 40 41 42 43 44 45 46 47 48 49 50 51 52 53 54 55 56 57 58 59 60 61 62 63 64 65 66 67 68 69 70 71 72 73 74 75 76 77 78 79 80 81 82 83 84 85 86 87 88 89 90 91 92 93 94 95 96 97 98 99 100

Process finished with exit code 0

**Tenth time running**

aaaaaaabbbbbbbbbbbbbbbbbbbbbbbbbbbbbbbbbbbbbbbbbbbbbbbbbbbbbbbbbbbbbbbbbbbbbbbbbbbbbbbbbbbbbbbbbbbbbbbbbbbbaaaaaa 1 2 3 4 5 6 7 8 9 10 11 12 13 14 15 16 17 18 19 20 21 22 23 24 25 26 27 28 29 30 31 32 33aaaaaaaaaaaaaaaaaaaaaaaaaaaaaaaaaaaaaaaaaaaaaaaaaaaaaaaaaaaaaaaaaaaaa 34aaaaaaaaaaaaaaaaaa 35 36 37 38 39 40 41 42 43 44 45 46 47 48 49 50cccccccccccccccccccccccccccccccccccccccc 51 52 53 54 55 56 57 58 59 60 61 62 63 64 65 66 67 68 69 70 71 72 73 74 75 76 77 78 79 80 81 82 83 84 85 86 87 88 89 90 91 92 93 94 95 96 97 98 99 100

Process finished with exit code 0

I see a variation when it comes to all letters, and I will list them below.

* With the A's they change position from being before the b's to being after the b's or in between them and sometimes mixed in numbers for example the tenth time running, there is a variation as it is after the numbers 33 and 34 as well as being in front of the number 1.
* With the B's they also change position much like the letter a. They have been recorded as being at the start before the a's, being after the a's or in between them but those are the only variations they are never recorded as being after a number.
* With the C's they have equal amounts of variation as A does apart from being at the start. It has been recorded that c's were after the numbers 48 and 50, as well as being in front of 1 after the a's and b's.

*Thread Program 1.3*

*Things to try part 1.3:*

*1. Create an Eclipse Java Project to use this program and run the software   
 10 times, each time copying and pasting your results into a word processor*

*Do you see much variation in your results from one run to the next? Explain  
 what conclusions you come to.*

*ANSWER(S)*

**First time running**

aaaaaaaaaaaaaaaaaaaaaaaaaaaaaaaaaaaaaaaaaaaaaaaaaaaaaaaaaaaabbbbbbbbbbbbbbbbbbbbbbbbbbbbbbbbbbbbbbbbbbbbbbbbbaaaaaaaaaaaaaaaaaaaaaaaaaaaaaaaaaaaaaaaabbbbbbbbbbbbbbbbbbbbbbbbbbbbbbbbbbbbbbbbbbbbbbbbbbb 1 2 3 4 5 6 7 8 9 10 11 12 13 14 15 16 17 18 19 20 21 22 23 24 25 26 27 28 29 30 31 32 33 34 35 36 37 38 39 40 41 42 43 44 45 46 47 48 49 50 51 52 53 54 55 56 57 58 59 60 61 62 63 64 65 66 67 68 69 70 71 72 73 74 75 76 77 78 79 80 81 82 83 84 85 86 87 88 89 90 91 92 93 94 95 96 97 98 99 100

Process finished with exit code 0

**Second time running**

aabbbbbbbbbbbbbbbbbbbbbbbbbbbbbbbbbbbbbbbbbbbbbbbbbbbbbbbbbbbbbbbbbbbbbbbbbbbbbbbbbbbbbbbbbbbbbbbbbbbbaaaaaaaaaaaaaaaaaaaaaaaaaaaaaaaaaaaaaaaaaaaaaaaaaaaaaaaaaaaaaaaaaaaaaaaaaaaaaaaaaaaaaaaaaaaaaaaaaa 1 2 3 4 5 6 7 8 9 10 11 12 13 14 15 16 17 18 19 20 21 22 23 24 25 26 27 28 29 30 31 32 33 34 35 36 37 38 39 40 41 42 43 44 45 46 47 48 49 50 51 52 53 54 55 56 57 58 59 60 61 62 63 64 65 66 67 68 69 70 71 72 73 74 75 76 77 78 79 80 81 82 83 84 85 86 87 88 89 90 91 92 93 94 95 96 97 98 99 100

Process finished with exit code 0

**Third time running**

aaaaaaaaaaaaaaaaaaaaaaaaaaaaaaaaaaaaaaaaaaaaaaaaaabbbbbbbbbbbbbbbbbbbbbbbbbbbbbbbbbbbbbbbbbbbbbbbbbbbbbbbbbbbbbbbbbbbbbbbbbbbbbbbbbbbbbbbbbbbbbbbbbbbb 1aaaaaaaaaaaaaaaaaaaaaaaaaaaaaaaaaaaaaaaaaaaaaaaaaa 2 3 4 5 6 7 8 9 10 11 12 13 14 15 16 17 18 19 20 21 22 23 24 25 26 27 28 29 30 31 32 33 34 35 36 37 38 39 40 41 42 43 44 45 46 47 48 49 50 51 52 53 54 55 56 57 58 59 60 61 62 63 64 65 66 67 68 69 70 71 72 73 74 75 76 77 78 79 80 81 82 83 84 85 86 87 88 89 90 91 92 93 94 95 96 97 98 99 100

Process finished with exit code 0

**Fourth time running**

aaaaaaaaaaaaaaaaaaaaaabbbbbbbbbbbbbbbbbbbbbbbbbbbbbbbbbbbbbbbbbbbbbbbbbbbbbbbbbbbbbbbbbbbbbbbbbbbbbbbbbbbbbbbbbbbbbbbbbbbbaaaaaaaaaaaaaaaaaaaaaaaaaaaaaaaaaaaaaaaaaaaaaaaaaaaaaaaaaaaaaaaaaaaaaaaaaaaaaa 1 2 3 4 5 6 7 8 9 10 11 12 13 14 15 16 17 18 19 20 21 22 23 24 25 26 27 28 29 30 31 32 33 34 35 36 37 38 39 40 41 42 43 44 45 46 47 48 49 50 51 52 53 54 55 56 57 58 59 60 61 62 63 64 65 66 67 68 69 70 71 72 73 74 75 76 77 78 79 80 81 82 83 84 85 86 87 88 89 90 91 92 93 94 95 96 97 98 99 100

Process finished with exit code 0

**Fifth time running**

abbbbaaaabbbbaabbbbbaaaaaaaaaaaaaaaaaaaaaaaaaabbbbbbbbbbbbbbaaaaaaaaaaaaaaaaaaaaaaaabbbbbbbbbbbbbbbbbbbbbbaaaaabbbbbbbbbbbbbbbbbbbbbbbbbbbbbbbbbbbbbbbbbbbbbbbaaaaaaaaaaaaaaaaaaaaaaaaaaaaaaaaaaaaaabbbb 1 2 3 4 5 6 7 8 9 10 11 12 13 14 15 16 17 18 19 20 21 22 23 24 25 26 27 28 29 30 31 32 33 34 35 36 37 38 39 40 41 42 43 44 45 46 47 48 49 50 51 52 53 54 55 56 57 58 59 60 61 62 63 64 65 66 67 68 69 70 71 72 73 74 75 76 77 78 79 80 81 82 83 84 85 86 87 88 89 90 91 92 93 94 95 96 97 98 99 100

Process finished with exit code 0

**Sixth time running**

aaabbbbbbbbbbbbbbbbbbbbbbbbbbbbbbbbbbbbbbbbbbbbbbbbbbbbbbbbbbbbbbbbbbbbbbbbbbbbbbbbbbbbbbbbbbbbbbbbbbbb 1aaaaaaaaaaaaaaaaaaaaaaaaaaaaaaaaaaaaaaaaaaaaaaaaaaaaaaaaaaaaaa 2aaaaaaaaaaaaaaaaaaaaaaaaa 3aaaaaaaaaa 4 5 6 7 8 9 10 11 12 13 14 15 16 17 18 19 20 21 22 23 24 25 26 27 28 29 30 31 32 33 34 35 36 37 38 39 40 41 42 43 44 45 46 47 48 49 50 51 52 53 54 55 56 57 58 59 60 61 62 63 64 65 66 67 68 69 70 71 72 73 74 75 76 77 78 79 80 81 82 83 84 85 86 87 88 89 90 91 92 93 94 95 96 97 98 99 100

Process finished with exit code 0

**Seventh time running**

bbbbbbbbbbbbbbbbbbbbaaaaaaaaabbbbbbbbbbbbbbbbbbbbbbbbbbbbbbbbbbbbbbbbbbbbbbbbbbbbbbbbbbbbbbbbbbbbbbbbbbbbbbbbaaaaaaaaaaaaaaaaaaaaaaaaaaaaaaaaaaaaaaaaaaaaaaaaaaaaaaaaaaaaaaaaaaaaaaaaaaaaaaaaaaaaaaaaaaa 1 2 3 4 5 6 7 8 9 10 11 12 13 14 15 16 17 18 19 20 21 22 23 24 25 26 27 28 29 30 31 32 33 34 35 36 37 38 39 40 41 42 43 44 45 46 47 48 49 50 51 52 53 54 55 56 57 58 59 60 61 62 63 64 65 66 67 68 69 70 71 72 73 74 75 76 77 78 79 80 81 82 83 84 85 86 87 88 89 90 91 92 93 94 95 96 97 98 99 100

Process finished with exit code 0

**Eighth time running**

abbbbbbbbbbbbbbbbbbbbbbbbbbbbbbbbbbbbbbbbbbbbbbbbbbbbbbbbbbbbbbbbbbbbbbbbbbbbbbbbbbbbbbbbbbbbbbbbbbbba 1aaaaaaaaaaaaaaaaaaaaaaaa 2aaaaaaaaaaaaaaaaaaaaaaaaaaa 3aaaaaaaaaaaaaaaaaaaaa 4aaaaaaaaaaaaaaaaaaaaaaaaaa 5 6 7 8 9 10 11 12 13 14 15 16 17 18 19 20 21 22 23 24 25 26 27 28 29 30 31 32 33 34 35 36 37 38 39 40 41 42 43 44 45 46 47 48 49 50 51 52 53 54 55 56 57 58 59 60 61 62 63 64 65 66 67 68 69 70 71 72 73 74 75 76 77 78 79 80 81 82 83 84 85 86 87 88 89 90 91 92 93 94 95 96 97 98 99 100

Process finished with exit code 0

**Ninth time running**

abbbbbbbbbbbbbbbbbbbbbbbbbbbbbbbbbbbbbbbbbbbbbbbbbbbbbbbbbbbbbbbbbbbbbbbbbbbbbbbbbbbbbbbbbbbbbbbbbbbbaaaa 1aaaaaaaaaaaaaaaaaaaaaaaaaaaaaaaaaaaaaaaaaaaaaaaaaaaaaaaaaaaaaaaaaaaaaaaaaaaaaaaaaaaaaaaaa 2aaaaaa 3 4 5 6 7 8 9 10 11 12 13 14 15 16 17 18 19 20 21 22 23 24 25 26 27 28 29 30 31 32 33 34 35 36 37 38 39 40 41 42 43 44 45 46 47 48 49 50 51 52 53 54 55 56 57 58 59 60 61 62 63 64 65 66 67 68 69 70 71 72 73 74 75 76 77 78 79 80 81 82 83 84 85 86 87 88 89 90 91 92 93 94 95 96 97 98 99 100

Process finished with exit code 0

**Tenth time running**

aaaaaaaaaaaaaaaabbbbbbbbbbbbbbbbbbbbbaaaaaaaaaaabbbbbbbbbbbbbbbbbbbbbbbbbbbbbbbbbbbbbbbbbbbbbbbbbbbbbbbbbbbbbbaaaaaaaaaaaaaaaaaaaaaaaaaaaaaaaaaaaaaaaaaaaaaaaaaaaaaaaaaaaaaaaaaaaaaaaaabbbbbbbbbbbbbbbbb 1 2 3 4 5 6 7 8 9 10 11 12 13 14 15 16 17 18 19 20 21 22 23 24 25 26 27 28 29 30 31 32 33 34 35 36 37 38 39 40 41 42 43 44 45 46 47 48 49 50 51 52 53 54 55 56 57 58 59 60 61 62 63 64 65 66 67 68 69 70 71 72 73 74 75 76 77 78 79 80 81 82 83 84 85 86 87 88 89 90 91 92 93 94 95 96 97 98 99 100

Process finished with exit code 0

*Thread Program 1.4*

*Things to try part 1.4:*

*1. Create a JavaEclipse Project to use this program and run the software 10 times,  
 each time copying and pasting your results into a word processor file. Label   
 this set of 10 results clearly with the heading Threading Program 1.4*

*Do you see much variation in your results from one run to the next? Explain  
 what conclusions you come to***.**

*ANSWER(S)*

**First time running**

aabbbbbbbaaaaaaaaaaaabbbbbbbbbbbbbbbaaaaaaaaaaaaaaaaaaaaaaaaaaaaaaaaaaaaaaaaaaaaaaaaaaaaaaaaaaaaaaaaaaaaaaaaaaaaaaaaaaaaaabbbbbbbbbbbbbbbbbbbbbbbbbbbbbbbbbbbbbbbbbbbbbbbbbbbbbbbbbbbbbbbbbbbbbbbbbbbbbb 1 2 3 4 5 6 7 8 9 10 11 12 13 14 15 16 17 18 19 20 21 22 23 24 25 26 27 28 29 30 31 32 33 34 35 36 37 38 39 40 41 42 43 44 45 46 47 48 49 50 51 52 53 54 55 56 57 58 59 60 61 62 63 64 65 66 67 68 69 70 71 72 73 74 75 76 77 78 79 80 81 82 83 84 85 86 87 88 89 90 91 92 93 94 95 96 97 98 99 100

Process finished with exit code 0

**Second time running**

abbbbbbbbbbbbbbbbbbbbbbbbbbbbbbbbbbbbbbbbbbbbbbbbbbbbbbbbbbbbbbbbbbbbbbbbbbbbbbbbbbbbbbbbbbbbbbbbbbbbaaa 1 2 3 4 5 6 7 8 9 10 11 12 13 14 15 16 17 18 19 20 21 22 23 24 25 26 27 28 29 30 31 32 33 34 35 36 37 38 39 40 41 42 43 44 45 46 47 48 49 50aaaaaaaaaaaaaaaaaaaaaaaaaaaaaaaaaaa 51aaaaaa 52aaaaaaaaaaaaaaaaaaa 53aaaaaaaaaaaaaaaaaaaaaaaaaaaaaaaaaaaa 54 55 56 57 58 59 60 61 62 63 64 65 66 67 68 69 70 71 72 73 74 75 76 77 78 79 80 81 82 83 84 85 86 87 88 89 90 91 92 93 94 95 96 97 98 99 100

Process finished with exit code 0

**Third time running**

aabbbbbbbbbbbbbbbaaaaaaaaaaaaaaabbbbbbbbbbbbbbbbbbbbbbbbbbbbbbbbbbaaaaaaaaaaaaaaaaabbbbbbbbbbbbbbbbbbbbaaaaaaaaaaaaaaaaaaaaaaaaaaabbbbbbbbbbbbbbbbbbbbbbbbaaaaaaaaaaaaaaaaaaaaaaaaaaaaaaaaaaaaaaabbbbbbb 1 2 3 4 5 6 7 8 9 10 11 12 13 14 15 16 17 18 19 20 21 22 23 24 25 26 27 28 29 30 31 32 33 34 35 36 37 38 39 40 41 42 43 44 45 46 47 48 49 50 51 52 53 54 55 56 57 58 59 60 61 62 63 64 65 66 67 68 69 70 71 72 73 74 75 76 77 78 79 80 81 82 83 84 85 86 87 88 89 90 91 92 93 94 95 96 97 98 99 100

Process finished with exit code 0

**Fourth time running**

bbbbbbbbbbbbbbbbbbbbbbbbbaaaaaaaaaaaaaaaaaaaaaaaaaaaaaaaaaaaaaaaaaaaaaaaaaaaaaaaaaaaaaaaaaaaaaaaaaaaaaaaaaaaaaaaaaaaaaaaaaaaabbbbbbbbbbbbbbbbbbbbbbbbbbbbbbbbbbbbbbbbbbbbbbbbbbbbbbbbbbbbbbbbbbbbbbbbbbb 1 2 3 4 5 6 7 8 9 10 11 12 13 14 15 16 17 18 19 20 21 22 23 24 25 26 27 28 29 30 31 32 33 34 35 36 37 38 39 40 41 42 43 44 45 46 47 48 49 50 51 52 53 54 55 56 57 58 59 60 61 62 63 64 65 66 67 68 69 70 71 72 73 74 75 76 77 78 79 80 81 82 83 84 85 86 87 88 89 90 91 92 93 94 95 96 97 98 99 100

Process finished with exit code 0

**Fifth time running**

abbbbbbbbbbbbbbbbbbbbbbbbbbbbbbbbbbbbbbbbbbbbbbaaaaaaaaaaaaaaaaaaaaaaaaaaaaaaaaaaaaaaaaaaaaaaaaaaaaaaaaaaaaaaaaaaaaaaaaaaaaaaaaaaaaaaaaaaaaaaaaaaabbbbbbbbbbbbbbbbbbbbbb 1 2 3 4 5 6 7 8 9 10 11 12 13 14 15 16 17 18 19 20 21 22 23 24 25 26 27 28 29 30 31 32 33 34 35 36 37 38 39 40 41 42 43 44 45 46 47 48 49 50bbbbbbbbbbbbbbbbbbbbbbbbbbbbbbbb 51 52 53 54 55 56 57 58 59 60 61 62 63 64 65 66 67 68 69 70 71 72 73 74 75 76 77 78 79 80 81 82 83 84 85 86 87 88 89 90 91 92 93 94 95 96 97 98 99 100

Process finished with exit code 0

**Sixth time running**

bbbbbbbbbbbbbbbbbbbbbbbbbbbbbbbbbbbbbbbbbbbbbbbbbbbaaaaaaaaaaaaaaaaaaaaaaaaaaaaaaaaaaaaaaaaaaaaaaaaaaaaaaaabbbbbbbbbbbbbbbbbbbbbbbbbbbbbbbbbbbbbbbbbbbbbbbbbaaaaaaaaaaaaaaaaaaaaaaaaaaaaaaaaaaaaaaaaaaaa 1 2 3 4 5 6 7 8 9 10 11 12 13 14 15 16 17 18 19 20 21 22 23 24 25 26 27 28 29 30 31 32 33 34 35 36 37 38 39 40 41 42 43 44 45 46 47 48 49 50 51 52 53 54 55 56 57 58 59 60 61 62 63 64 65 66 67 68 69 70 71 72 73 74 75 76 77 78 79 80 81 82 83 84 85 86 87 88 89 90 91 92 93 94 95 96 97 98 99 100

Process finished with exit code 0

**Seventh time running**

bbbbbbbbbbbbbbbbbbbbbbbbbbbbbbbbbbbbbbbbbbbbbbbbbbbbbbbbbbbbbbbbbbbbbbbbbbbbbbbbbbbbbbbbbbbbbbbbbbbbaaaaaaaaaaaaaaaaaaaaaaaaaaaaaaaaaaaaaaaaaaaaaaaaaaaaaaaaaaaaaaaaaaaaaaaaaaaaaaaaaaaaaaaaaaaaaaaaaaaa 1 2 3 4 5 6 7 8 9 10 11 12 13 14 15 16 17 18 19 20 21 22 23 24 25 26 27 28 29 30 31 32 33 34 35 36 37 38 39 40 41 42 43 44 45 46 47 48 49 50 51 52 53 54 55 56 57 58 59 60 61 62 63 64 65 66 67 68 69 70 71 72 73 74 75 76 77 78 79 80 81 82 83 84 85 86 87 88 89 90 91 92 93 94 95 96 97 98 99 100

Process finished with exit code 0

**Eighth time running**

abbbbbbbbbbbbbbbbbbbbbbbbbbbbbbbbbbbbbaaaaaaaaaaaaaaaaaaaaaaaaaaaaaaaaaaaaaaaaaaaaaaaaaaaaaaaaaaaaaaaaaaaaaaaaaaaaaaaaaaaaaaaaaaaaaaaaaaabbbbbbbbbbbbbbbbbbbbbbbbbbbbbbbbbbbbbbbbbbbbbbbbbbbbbbbbbbbbbbb 1 2 3 4 5 6 7 8 9 10 11 12 13 14 15 16 17 18 19 20 21 22 23 24 25 26 27 28 29 30 31 32 33 34 35 36 37 38 39 40 41 42 43 44 45 46 47 48 49 50 51 52 53 54 55 56 57 58 59 60 61 62 63 64 65 66 67 68 69 70 71 72 73 74 75 76 77 78 79 80 81 82 83 84 85 86 87 88 89 90 91 92 93 94 95 96 97 98 99 100

Process finished with exit code 0

**Ninth time running**

bbbbbbbbbbbbbbbbbbbbbbbbbbbbbbbbbbbbbbbbbbbbbbbbbbbbbbbaaaaaaaaaaaaaaaaaaaaaaaaaaaaaaaaaaaaaaaaaaaaaaaaaaaaaaaaaaaaaaaaaaaaaaaaaaaaaaaaaaaaaaaaaaaaaaaaaaaabbbbbbbbbbbbbbbbbbbbbbbbbbbbbbbbbbbbbbbbbbbbb 1 2 3 4 5 6 7 8 9 10 11 12 13 14 15 16 17 18 19 20 21 22 23 24 25 26 27 28 29 30 31 32 33 34 35 36 37 38 39 40 41 42 43 44 45 46 47 48 49 50 51 52 53 54 55 56 57 58 59 60 61 62 63 64 65 66 67 68 69 70 71 72 73 74 75 76 77 78 79 80 81 82 83 84 85 86 87 88 89 90 91 92 93 94 95 96 97 98 99 100

Process finished with exit code 0

**Tenth time running**

abbbbbbbbbbbbbbbbbbbbbbbbbbbbbbbbbbbbbbbbbbbbbbbbbbbbbbbbbbbbbbbbbbbbbbbbbbbbbbbbbbbbbbbbbbbbbbbbbbbbaaaaaaaaaaaaaaaaaaaaaaaaaaaaaaaaaaaaaaaaaaaaaaaaaaaaaaaaaaaaaaaaaaaaaaaaaaaaaaaaaaaaaaaaaaaaaaaaaaa 1 2 3 4 5 6 7 8 9 10 11 12 13 14 15 16 17 18 19 20 21 22 23 24 25 26 27 28 29 30 31 32 33 34 35 36 37 38 39 40 41 42 43 44 45 46 47 48 49 50 51 52 53 54 55 56 57 58 59 60 61 62 63 64 65 66 67 68 69 70 71 72 73 74 75 76 77 78 79 80 81 82 83 84 85 86 87 88 89 90 91 92 93 94 95 96 97 98 99 100

Process finished with exit code 0

there are three different variations I noticed when I ran this three times and they are the following:

* The a's and b's switch around before the 1 numerous of times.
* In the Second running results there were a's after the numbers 51 , 52 and 53.
* In the Fifth running results there were b's after the number 50.

*2. Repeat part one above several times, each time with a different sleep time.  
 Write out your conclusions*

*ANSWER(S)*

**Thread.sleep(10);**

abbbbbbbbbbbbbbbbbbbbbbbbbbbbbbbbbbbbbbbbbbbbbbbbbbbbbbbbbbbbbbbbbbbbbbbbbbbbbbbbbbbbbbbbaaaaaaaaaaaaaaaaaaaaaaaaaaaaaaaaaaaaaaaaaaaaaaaaaaaaaaaaaaaaaaaaaaaaaaaaaaaaaaaaaaaaaaaaaaaaaaaaaaabbbbbbbbbbbb 1 2 3 4 5 6 7 8 9 10 11 12 13 14 15 16 17 18 19 20 21 22 23 24 25 26 27 28 29 30 31 32 33 34 35 36 37 38 39 40 41 42 43 44 45 46 47 48 49 50 51 52 53 54 55 56 57 58 59 60 61 62 63 64 65 66 67 68 69 70 71 72 73 74 75 76 77 78 79 80 81 82 83 84 85 86 87 88 89 90 91 92 93 94 95 96 97 98 99 100

Process finished with exit code 0

**Thread.sleep(20);**

aaaaaaaaaaaaaaaaaaaaaaaaaaabbbbbbbbbaaaaaaaaaaaaaaaaaaaaaaaaaaaaaaaaaaaaaaaaaaaaaaabbbbbbbbbbbbbbbbbbbbbbbbbbbbbbbbbbbbbbbbbbbbbbaaaaaaaaaaaaaaaaaaaaaaaaabbbbbbbbbbbbbbbbbbbbbbbbbbbbbbbbbbbbbbbbbbbbba 1 2 3 4 5 6 7 8 9 10 11 12 13 14 15 16 17 18 19 20 21 22 23 24 25 26 27 28 29 30 31 32 33 34 35 36 37 38 39 40 41 42 43 44 45 46 47 48 49 50 51 52 53 54 55 56 57 58 59 60 61 62 63 64 65 66 67 68 69 70 71 72 73 74 75 76 77 78 79 80 81 82 83 84 85 86 87 88 89 90 91 92 93 94 95 96 97 98 99 100

Process finished with exit code 0

**Thread.sleep(30);**

bbbbbbbbaaaaaaaaaaaaaaaaaaaaaaaaaaaaaaaaaaaaaaaaaaaaaaaaaaaaaaaaaaaaaaaaaaaaaaaaaaaaaaaaaaaaaaaaaaaaaaaaaaaabbbbbbbbbbbbbbbbbbbbbbbbbbbbbbbbbbbbbbbbbbbbbbbbbbbbbbbbbbbbbbbbbbbbbbbbbbbbbbbbbbbbb 1 2 3 4 5 6 7 8 9 10 11 12 13 14 15 16 17 18 19 20 21 22 23 24 25 26 27 28 29 30 31 32 33 34 35 36 37 38 39 40 41 42bbbbbbb 43 44 45 46 47 48 49 50 51 52 53 54 55 56 57 58 59 60 61 62 63 64 65 66 67 68 69 70 71 72 73 74 75 76 77 78 79 80 81 82 83 84 85 86 87 88 89 90 91 92 93 94 95 96 97 98 99 100

Process finished with exit code 0

**Thread.sleep(40);**

aaaaaaaaaaaaaaaaaaaaaaaaaaaaaaaaaaaaaaaaaaaaaaaaaaaaaaaaaaaaaaaaaaaaaaaaaaaaaaaaaaaabbbbbbbbbbbbbbbbbbbbbbbbbbbbbbbbbbbbbbbbbbbbbbbbbbbbbbbbbbbbbbbbbbbbbbbbbbbbbbbbbbbbbbbbbbbbbbbbbbbbaaaaaaaaaaaaaaaa 1 2 3 4 5 6 7 8 9 10 11 12 13 14 15 16 17 18 19 20 21 22 23 24 25 26 27 28 29 30 31 32 33 34 35 36 37 38 39 40 41 42 43 44 45 46 47 48 49 50 51 52 53 54 55 56 57 58 59 60 61 62 63 64 65 66 67 68 69 70 71 72 73 74 75 76 77 78 79 80 81 82 83 84 85 86 87 88 89 90 91 92 93 94 95 96 97 98 99 100

Process finished with exit code 0

The code *thread.sleep()*; causes the thread to suspend its execution for the specified amount of time which is for example 10,20,30,40.

*Things To Try 1.5*

*Things to try part 1.5:*

1. *Consider now the program for part 1.1 above and, in particular, the following*

*lines:*

*// Third code block: Start threads*

*thread1.start();*

*thread2.start();*

*thread3.start();*

*Now change these lines to read:*

*// Third code block: Start threads*

*thread1.start();*

*thread2.start();*

*thread3.setPriority(Thread.MAX\_PRIORITY);*

*thread3.start();*

*Do the first of the things to do of part 1.1 (see page 3) again but using the modified program this time.*

*Compare your results with those that you got the first time.*

*ANSWER(S)*

**First time running**

bbbbbbbbbbbbbbbbbbbbbbaaaaaaaaaaaaaaaaaaaaaaaaaaaaaaaaaaaaaaaaaaaaaaaaaaaaaaaaaaaaaaaaaaaaaaaaaaaaaaaaaaaaaaaaaaaaaaaaaaaabbbbbbbbbbbbbbbbbbbbbbbbbbbbbbbbbbbbbbbbbbbbbbbbbbbbbbbbbbbbbbbbbbbbbbbbbb 1b 2bbb 3 4 5 6 7 8 9 10 11 12 13 14 15 16 17 18 19 20 21 22 23 24 25 26 27 28 29 30 31 32 33 34 35 36 37 38 39 40 41 42 43 44 45 46 47 48 49 50 51 52 53 54 55 56 57 58 59 60 61 62 63 64 65 66 67 68 69 70 71 72 73 74 75 76 77 78 79 80 81 82 83 84 85 86 87 88 89 90 91 92 93 94 95 96 97 98 99 100

Process finished with exit code 0

**Second time running**

aaaaaaabbbbbbbbbbbbbbbbbbbbbbbbbbbbaaaaaaaaaaaaaaaaaaaaaaaaaaaaaaaaaaaaaaaaaaaaaaaaaaaaaaaaaaaaaaaaaaaaaaaaaaaaaaaaaaaaaaaaaaaaabbbbb 1 2 3 4 5 6 7 8 9 10 11 12 13 14 15bbbbbb 16b 17 18 19 20 21 22 23 24 25 26 27 28 29 30 31 32 33 34 35 36 37 38 39 40 41 42 43 44 45 46 47 48 49 50bbbbbbbbbbbbbbbbbbbbbbbbbbbbbbbbbbbbbbbbbbbbbbbbbbbbbbbbbbbb 51 52 53 54 55 56 57 58 59 60 61 62 63 64 65 66 67 68 69 70 71 72 73 74 75 76 77 78 79 80 81 82 83 84 85 86 87 88 89 90 91 92 93 94 95 96 97 98 99 100

Process finished with exit code 0

**Third time running**

aaaaaaaaaaaaaaaaaaaaaaaaaaaaaaaaaaaaaaaaaaaaaaaaaaaaaaaaaaaaaaa 1 2 3 4 5 6 7 8 9 10 11 12 13 14 15 16 17 18 19 20 21 22 23 24 25 26 27 28 29 30 31 32 33 34 35 36 37 38 39 40 41 42 43 44 45 46 47 48 49 50aaaaaaaaaaaaaaaaaaaaaaaaaaaaaaaaaa 51aaabbbbbbbbbbbbbbbbbbbbbbbbbbbbbbbb 52bbbbbbbbbbbbbbbbbbbbbbbbbbbbbbb 53bbbbbbbbbbbbbbbbbbbbbbbbbbbbbbbbbbbbb 54 55 56 57 58 59 60 61 62 63 64 65 66 67 68 69 70 71 72 73 74 75 76 77 78 79 80 81 82 83 84 85 86 87 88 89 90 91 92 93 94 95 96 97 98 99 100

Process finished with exit code 0

**Fourth time running**

abbbbbbbbbbbbbbbbbbbbbbbbbbbbbbbbbbbbbbbbbbbbbbbbbbbbbbbbbbaaaaaaaaaabbbbbbbbbbbbbbbbbbbbbbbbbbbbbbbbbbbbbbbbbbaaaaaaaaaaaaaaaaaaaaaaaaaaaaaaaaaaaaaaaaaaaa 1 2 3 4 5 6 7 8 9 10 11 12 13 14 15 16 17 18 19 20 21 22 23 24 25 26 27 28 29 30 31 32 33 34 35 36 37 38 39 40 41 42 43 44a 45 46 47 48 49 50aaaaaaaaaaaaaaaaaaaaaaaaaa 51aaaaaaaaaaaaaaaaaa 52 53 54 55 56 57 58 59 60 61 62 63 64 65 66 67 68 69 70 71 72 73 74 75 76 77 78 79 80 81 82 83 84 85 86 87 88 89 90 91 92 93 94 95 96 97 98 99 100

Process finished with exit code 0

**Fifth time running**

bbbbbbbbbbbbbbbbbbbbbbbbbbbbbbbbbbbbbbbbbbbbbbbbbbbbbbbbbbbbbbbbbbbbbbbbbbbbbbbbbbbbbbbbbbbbbbbbbbbb 1 2 3 4 5 6 7 8 9 10 11a 12 13 14 15 16 17 18 19 20 21 22 23 24 25 26 27 28 29 30 31 32 33 34 35 36 37 38 39 40 41 42 43 44 45 46 47 48 49 50aaaaaaaaaaaaaaaaaaaaaaaaaaaaaaaaaaaaaaaaaaaa 51aaaaaaaaaaaaaaaaaaaaaaaaaaaaaaaaaaaaaaaaaaaaaaaaaaaaaaa 52 53 54 55 56 57 58 59 60 61 62 63 64 65 66 67 68 69 70 71 72 73 74 75 76 77 78 79 80 81 82 83 84 85 86 87 88 89 90 91 92 93 94 95 96 97 98 99 100

Process finished with exit code 0

**Sixth time running**

bbbbbbbbbbbbbbbbbbbbbbbbbbbbbbbbbbbbbbbbbbbbbbbbbbbbbbbbbbbbbbbbbbbbbbbbbbbbbbbbbbbbbbbbbbbbbbbbbbbbaaaaaaaaaaaaaaaaaaaaaaaaaaaaaaaaaaaaaaaaaaaaaaaaaaaaaaaaa 1 2 3 4 5 6 7 8 9 10 11 12 13 14 15 16 17 18 19 20 21 22 23 24 25 26 27 28a 29 30 31 32 33 34 35 36 37 38 39 40 41 42 43 44 45 46 47 48 49 50aaaaaaaaaaaaaaaaaaaaaaaaaaaaaaaaaaaaaaaaaa 51 52 53 54 55 56 57 58 59 60 61 62 63 64 65 66 67 68 69 70 71 72 73 74 75 76 77 78 79 80 81 82 83 84 85 86 87 88 89 90 91 92 93 94 95 96 97 98 99 100

Process finished with exit code 0

**Seventh time running**

bbbbbbbbbbbbbbbbbbbbbbbbbbbbbbbbbbbbbaaaaaaaaaaaaaaaaaaaaabbbbbbbbbbbbbbbbbbbbbbbbbbbbbbbbbbbbbbbbbbbbbbbbbbbbbbbbbbbbbbbaaaaaaaaaaaaa 1 2 3 4 5 6 7 8 9 10 11 12 13 14 15 16 17 18 19 20 21 22 23 24 25 26 27 28 29 30 31 32 33 34 35 36 37 38 39 40 41 42 43 44 45 46 47 48 49 50aaaaaaaaaaaaaaaaaaaaaaaaaaaaaaaaaaaaaaaaaaaaaaaa 51aaaaaaaaaaaaaaaaaa 52 53 54 55 56 57 58 59 60 61 62 63 64 65 66 67 68 69 70 71 72 73 74 75 76 77 78 79 80 81 82 83 84 85 86 87 88 89 90 91 92 93 94 95 96 97 98 99 100

Process finished with exit code 0

**Eighth time running**

abbbbbbbbbbbbbbbbbbbbbbbbbbbbbbbbbbbbbbbbbbbbbbbbbbbbbbbbbbbbbbbbbbbbbbbbbbbbbbbbbbbbbbbbbbbbbbbbbbbbaaaaaaaaaaaaaaaaaaaaaaaaaaaaaaaaaaaaaaaaaaaaaaaaaaaaaaaaaaaaaaaaaaaaaaaaaaaaaaaaa 1 2 3 4 5 6 7 8 9 10 11 12 13 14 15 16 17 18 19 20 21 22 23 24 25 26 27 28 29 30 31 32 33 34 35 36 37 38 39 40 41 42 43 44 45 46 47 48 49 50aaaaaaaaaaaaaaaaaa 51 52 53 54 55 56 57 58 59 60 61 62 63 64 65 66 67 68 69 70 71 72 73 74 75 76 77 78 79 80 81 82 83 84 85 86 87 88 89 90 91 92 93 94 95 96 97 98 99 100

Process finished with exit code 0

**Nineth time running**

aabbbbbbbbbbbbbbbbbbbbbbbbbbbabbbbbbbbbbbbbbbbbbbbbbbbbbbbbbbbbbbbbbbbbbbbbbbbbbbbb 1 2 3 4 5 6 7 8aaaaaaaaaaaaaaaaaaaaaaaaaaaaaaaaaaaaaaaaaaaaaaaaaaaaaabbbbbbbb 9 10 11 12 13 14 15 16 17 18 19 20 21 22 23 24 25 26 27 28 29 30 31 32 33 34 35 36 37 38 39 40 41 42 43 44 45 46 47 48 49 50aaaaaaaaaaaaaaaaaaaaaaaaaaaaaaaaaaaaaaaaaaabbbbbbbbb 51bbb 52 53 54 55 56 57 58 59 60 61 62 63 64 65 66 67 68 69 70 71 72 73 74 75 76 77 78 79 80 81 82 83 84 85 86 87 88 89 90 91 92 93 94 95 96 97 98 99 100

Process finished with exit code 0

**Tenth time running**

aaaaaaaaabbbbbbbbbbbbbbbbbbbbbbbbbbbbbbbbbbbbbbbbbbbbbbbbbbbbbbbbbbbbbbbbbbbbbbbb 1 2 3 4 5 6 7 8 9 10 11 12 13 14 15 16 17 18 19 20 21 22 23 24 25 26 27 28 29 30 31 32 33 34 35 36 37 38 39 40 41 42 43 44 45 46 47 48 49 50bbbbbaaaaaaaaabbbbbbbbbbbbbbbbbbbbbbbaa 51aaaaaaaaaaaaaaaaaaaaaaaaaa 52aaaaaaaaaaaaaaaaaaaaaaaaaaaaaaaaaaaaa 53aaaaaaaaaaaaaaaaa 54 55 56 57 58 59 60 61 62 63 64 65 66 67 68 69 70 71 72 73 74 75 76 77 78 79 80 81 82 83 84 85 86 87 88 89 90 91 92 93 94 95 96 97 98 99 100

Process finished with exit code 0

**These are the following variations of the program results:**

* **In the first**, we had the order b's, a's b's before 1 which can also be seen in the second running time.
* **In the second running time**, the order was the same as the first running time before the one, but the results also showed the numbers 15 and 50 with b's after them.
* **In the third running time**, the order at the start was as follows only a's at the start and had the following results 50 with numerous a's after, 51 with both a's and b's after, 52 with b's after and 53 with b's after the number also.
* **In the fourth running time**, the order at the start is as shown a's, b's, a's, b's before the 1 and had the following results 50 had numerous a's after the number and the same was seen with the number 51.
* **In the fifth running time**, the order at the start is as documented which had only b's and had the following results of the numbers 50 and 51 having a's after the number.
* **In the sixth running time**, the order at the beginning of the results was, b's and then there is before the number one and had similar results to the fifth running time, but the number 51 was left alone.
* **In the seventh running time**, the order before the number 1 was documented as b's, a's, b's, a's and had the same results as the fifth running time.
* **In the eighth-running time**, the order before the number 1 was found as a's b's a's and the number 50 had the letter a attached to it.
* **In the ninth running time**, the order of the letters before the 1 resulted with a's and b's the number 8 had the letters a and b attached to it and the number 50 also had the letter a and b attached to it.
* **In the tenth running time**, the following order was the same as the ninth running time and the following numbers had numerous letters attached to them, 50 b a b a, 51 a,52 a, 53 a.

**Compared to the Thread Program 1.1**

In comparison to the **Thread Program 1.1** In this program results above this extract firstly there were **no c's** whereas in Thread Program 1.1, there are c's and in this particular version of the program **the letters interfere with the numbers** whereas the Thread Program 1.1 only had the letters either at the start or the end as a result of when the program was run.

*Things To Try 2*

*Things to try part 2:*

*When you’ve run the program above write a paragraph that explains what you see in terms of what you’ve learned in lectures about*

1. *buffers,*
2. *multiprocessing/multithreading*
3. *the producer-consumer problem.*

*ANSWER(S)*

***Buffers:***

A buffer is a part of memory put aside for temporary data storage in which other data is being moved from one place to another. In java a buffer contains data for a specific primitive type. In this case it is used to store data/value given by the **ProducerTask** by using the code **buffer.write(i++);** the value is written into the buffer and is then stored only if the information meets the specific primitive requirements.

***Multiprocessing/multithreading:***

**Multiprocessing = multiple CPU’s**

**Multithreading = multiple threads per process**

In the program written, there is an import called import java.util.concurrent.\*;  
import java.util.concurrent.locks.\*; these are linked to multithreading. This package provides a tool to handle concurrency and the execution of threads. these two imports are very useful when it comes to concurrent programming and provide classes with helpful functionality.

***The Producer/Consumer Issue:***

This issue is one in a collection of concurrent programming complications which has the following details:

It has a finite-size buffer and two classes of threads, producers, and consumers, that place items into the buffer which is done by the producer and is then taken out by the consumer.

The problem here is that if the producer is waiting to put something in the consumer cannot take anything out.

*Newest ‘buffer.’ questions* (no date) *Stack Overflow*. Available at: https://stackoverflow.com/questions/tagged/buffer. (Accessed: 16 November 2023).

(No date) *The producer/consumer problem (multithreaded programming guide)*. Available at: https://docs.oracle.com/cd/E19455-01/806-5257/sync-31/index.html#:~:text=This%20problem%20is%20one%20of,of%20the%20buffer%20(consumers). (Accessed: 16 November 2023).

*Things To Try 3*

*Things to try part 3:*

1. *Run the above program, experimenting with different sleep times, than 100ms. Try, for example, 1ms, 20ms, 50ms, 200 ms, 300ms.*

*ANSWER(S)*

***Sleep time 1ms***

A screenshot of a computer program

Description automatically generated A screen shot of a computer program

Description automatically generated

***Sleep time 20ms***

A screenshot of a computer

Description automatically generated A screen shot of a computer

Description automatically generated

***Sleep time 50ms***

A screenshot of a computer

Description automatically generated A screen shot of a computer

Description automatically generated\

***Sleep time 200ms***

***A screenshot of a computer

Description automatically generated*** ***A screen shot of a computer

Description automatically generated***

***Sleep time 300ms***

***A screenshot of a computer program

Description automatically generated*** ***A screen shot of a computer

Description automatically generated***

Sleep time method is used to pause the execution of the thread in milliseconds. The times I have tested were *1ms,20ms,50ms,200ms,300ms.* As the sleep time increased the speed the thread executed became slower.

*2. Now that you’ve run the program, can you see clearly (not from the code but*

*simply from what you observe when running it) why this program, is simple,*

*though it is, cannot be implemented with only one thread? Explain.*

*3. Now eliminate both the try block and the catch block so that the inside of*

*the while block just looks like:*

*counterWindow.setText("" + count);*

*count++;*

*if(count > 10)*

*{*

*count=1;*

*}*

*After modification, run the program again and explain what you see.*

*ANSWER(S)*

A screen shot of a computer program

Description automatically generated A screenshot of a computer

Description automatically generated

There is no sleep time so now the speed of the thread is significantly faster than what I had previously at *200ms.* The if statement says the count of the number is greater than 10 then the count is made equal to one. Compared to the previous code with the *try* and *catch* it contained the sleep thread method referred to as *cThread* which enables you to set the sleep to whatever milliseconds suit as in this new code you have no choice.

*Things To Try 4*

*Things to try part 4*

*Consider the following software:*

*RunnableJob.java*

*package com.cakes;*

*public class RunnableJob implements Runnable {*

*@Override*

*public void run() {*

*Thread thread = Thread.currentThread();*

*System.out.println("RunnableJob is being run by " +*

*thread.getName() + " (" + thread.getId() + ")");*

*}*

*}*

*ThreadExample.java*

*package com.cakes;*

*public class ThreadExample {*

*public static void main(String[] args)*

*throws InterruptedException*

*{*

*RunnableJob runnableJob = new RunnableJob();*

*Thread thread1 = new Thread(runnableJob);*

*thread1.setName("thread1");*

*thread1.start();*

*Thread thread2 = new Thread(runnableJob, "thread2");*

*thread2.start();*

*Thread thread3 = new Thread(runnableJob);*

*thread3.start();*

*Thread currentThread = Thread.currentThread();*

*System.out.println("Main thread: " +*

*currentThread.getName() + "(" +*

*currentThread.getId() + ")");*

*}*

*}*

*Edit the software above and instead of using getName() and getID() use:*

*thread = Thread.currentThread();  
System.out.println(thread);*

*The last line will make thread use the Thread class's toString method.*

*Write down what you see.*

*ANSWER(S)*

**Before changing the code**

**A screenshot of a computer

Description automatically generated**

**After changing the code**

**A screen shot of a computer

Description automatically generated**

What is happening here is that there is RunnbleJob class and has two threads that run concurrently, thread 2 is run and takes up a resource so instead of there being 24 there are 23. The same happens when thread 1 is run it takes up a resource and now instead of 23, there are 22. When the threads are done running the resources are free again and go back to 24 available spaces.

END OF PRATICAL 7

END OF SECTION

END OF PORTFOLIO