Multithreading

DESCRIPTION

1. Getopt()

The getopt() function is a command-line parser that can be used by applications that follow Utility Syntax Guidelines. The syntax of getopt() is

```
getopt(int argc, char *const argv[], const char *optstring)
```

The parameters argc and argv are the argument count and argument array as passed to main(). The argument optstring is a string of option characters; if a character is followed by a colon, the option takes an argument.

Generally, the getopt() function is called from inside of a loop's conditional statement. The loop terminates when the getopt() function returns -1. A switch statement is then executed with the value returned by getopt() function.

Below is an example to use getopt() in the program race.java.

```
public static void main(String[] args) {
    // parse command line arguments, if any, to override defaults
    GetOpt go = new GetOpt(args, "UtM:");
    go.optErr = true;
    String usage = "Usage: -t -M m";
    int ch = -1;
    boolean timeSlicingEnsured = false;
    while ((ch = go.getopt()) != go.optEOF) {
      if     ((char)ch == 'U') {
        System.out.println(usage);  System.exit(0);
      }
      else if ((char)ch == 'M')
        M = go.processArg(go.optArgGet(), M);
      else {
        System.err.println(usage);  System.exit(1);
      }
    }

...
}
```

2. Template to write multithreading programs using Runnable interface in Java

```
(1)  class Worker implements Runnable { // implement the Runnable interface
             …
             public void run(){    // implement the abstract function
```

```
                    ...
                }
            }

    class mainThreads {

            ...

            //create a thread function object

            Worker workerObject = new Worker("workerObject", M);

            // create an array of threads

            Thread[] workers = new Thread[num];

              for (int i = 0; i < num; i++)

                  // create num threads using loop

                    workers[i] = new Thread(workerObject, "Thread" + i);

               for (int i = 0; i < num; i++) {

                    workers[i].start();  // start each thread

               }

            ...

    }
```

Hints: If there is not java installed on the machine, you can install and check java using commands below.

```
sudo apt update

sudo apt install default-jre

sudo apt install default-jdk

javac -version
```

GOALS

1. Learn how GetOpt utility works.

2. Learn how to write multithread program in java.

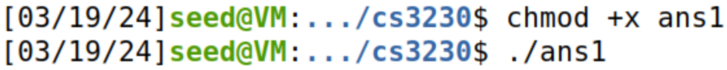
TASKS

Task1: Generate your first script file (ask about this in class) giving the following commands:

(1) Script File Name: ans1

```
javac GetOpt.java

java GetOpt -U

java GetOpt -aaa -b true -f theFile -w -80 -h3.33 arg1 arg2

java GetOpt -aaa -x -w90

java GetOpt -aaaaa -w90

java GetOpt -af theFile -w8 -b true

exit
```

(2) Run the shell script file ans1:



```
[03/19/24]seed@VM:.../cs3230$ chmod +x ans1
[03/19/24]seed@VM:.../cs3230$ ./ans1
```

-- **Make sure** to look at and understand this test program and the results, ask if you have questions.

Task2:

Task 2.1: Fill in blanks in the source code race.java**.**

Task 2.2: Generate your second script file giving the following commands:

Script File Name: ans2

```
javac *.java

java RaceTwoThreads -U

java RaceTwoThreads

java RaceTwoThreads -M50

java RaceTwoThreads -M70

exit
```

SUBMISSION

1. A report with the results of script 1: ans1 and script 2: ans2 and detailed explanations of the results in script 1: ans1 and script 2: ans2.
2. The complete source code of race.java.

Rubric

Criteria	Ratings			Pts
Execute: Compile and execute correctly with no errors	**10 pts** Correct	**0 pts** Not correct or no submission		10 pts
Script1: The result is correct.	**10 pts** Correct	**0 pts** Not correct or no submission		10 pts
Script1 explanation: Explanation of the results.	**20 pts** Correct	**10 pts** Some mistakes	**0 pts** Not correct or no submission	20 pts
Script2: The result is correct.	**10 pts** Correct	**0 pts** Not correct or no submission		10 pts
Script2 explanation: Explanation of the results.	**20 pts** Correct	**10 pts** Some mistakes	**0 pts** Not correct or no submission	20 pts
race.java: Blanks are filled correctly.	**30 pts** Correct	**20 pts** Some errors	**0 pts** Not correct or no submission	30 pts
Total Points: 100				

CONGRATULATIONS, YOU'VE COMPLETED PROGRAM 3!