Laboratory 3

At this laboratory, we learned about concurrent computation using thread scheduler.In java thread scheduler is a parh of the Java Virtual Machine that decides which thread should run. An important thing to know is that only one thread can run on a single process at a time and there is no guarantee on which runnable thread will be chosen to run by the thread scheduler.

We had some tasks for this lab:

Task 1:In this task I used the code which was here and the output will be if I use Start method, like this:1 1, 2 2 , 3 3 ,.. we will do that 10 times and we have this because we start both threads in the same time so the program will put that output.

Task 2: Hereafter we change the start method with the run method the output will be like:1 , 2 ,3 ,4….. and 1 , 2 , 3 ,4 …. So with this method we will do them on in the queue and not in parallel, first thread will print first and after him the second thread

For task 3 unfortunately I don't know how to write something.

Task 4:For this task from what I've seen in the course we have 2 methods there and I think the second one is more efficient because we can make more threads for this interval and every thread will get some part in this interval. Also we can eliminate multiple who are greater than that number from that interval because they are not prime numbers and this will make much more efficient and easier I think.That's what I deducted from them

Task 5:In this task from what I saw main will wait for all the horses to make their moves, they make some moves and after they make their moves, the main will print the result but the result can be different even if I have the same sleep time. So 1 time horse number 7 can win and and at the next simulation can be number 5.That's apply if we increase sleep time also.

```java
public class HorseThread extends Thread {
    private int horseNumber;
    private int numberOfMoves = 0;
    private boolean exit = false;
    public void setNumber(int number) {
        horseNumber = number;
    }
    public void run() {
        while(!exit) {
            if(numberOfMoves == 20) {
                System.out.println("Horse " + horseNumber + " finished.");
                exit = true;
            }
            else {
                numberOfMoves++;
                System.out.println("Horse " + horseNumber + "made a move.");
                try {
                    Thread.sleep(10);
                } catch (InterruptedException e) {
                    e.printStackTrace();
                }
            }
        }
    }
}
```

```java
package horse;

import java.util.Vector;

public class main{
public static void main(String[] args) {
    Vector<HorseThread> horseList = new Vector<HorseThread>();

    for(int i = 0; i < 20; i++) {
        horseList.add(new HorseThread());
    }

    for(int i = 0; i < 20; i++) {
        horseList.get(i).setNumber(i);
    }

    for(int i = 0; i < 20; i++) {
        horseList.get(i).start();
    }

    for(int i = 0; i < 20; i++) {
        try {
            horseList.get(i).join();
        } catch (InterruptedException e) {
            e.printStackTrace();
        }
    }
}

}
```

Task 6: From what I saw on internet there are many methods besides those mentioned. Some of them are: Interrupted this test whether the current thread has been interrupted.The interrupted status of the thread is cleared by this method.We have Isinterrupted and here the status is unaffected. Destroy this method was designed to destroy this thread without any cleanup. isAlive test if the thread is alive . setPriority changes the priority of the tread and also a stop but this method is inherently unsafe.