Basics of Multithreading part-1

Assignment

1) Create and Run a Thread using Runnable Interface and Thread class and show usage of sleep and join methods in the created threads.

```
public class Mythread extends Thread{
   private String threadname;
   public Mythread(String threadname){
       this.threadname=threadname;
   }
   @Override
   public void run(){
       System.out.println(threadname+ " running.");
       for(int i=0;i<5;i++){
           System.out.println(threadname+" running for "+i+"th time.");
       }
       try{
           Thread.sleep(1000);
       }
       catch(InterruptedException e){
           System.out.println("Exception occurred");
       }
       System.out.println(threadname+" after sleep");

   }
}
public class RunableThread implements Runnable{
   private String threadname;
   public RunableThread(String threadname){
       this.threadname=threadname;
   }
   @Override
   public void run(){
       System.out.println(threadname+ " running.");
       for(int i=0;i<5;i++){
           System.out.println(threadname+" running for "+i+"th time.");
       }
       try{
           Thread.sleep(1000);
       }
       catch(InterruptedException e){
           System.out.println("Exception occurred");
       }
       System.out.println(threadname+" after sleep");

   }
}
public class Main {
```

```java
    public static void main(String[] args) throws InterruptedException {
        Thread th=new Mythread("Thread1");
        Runnable th2=new RunableThread("thread2");
        Thread thr=new Thread(th2);
        th.start();
        thr.start();
        th.join();
        thr.join();
        System.out.println("Threads have finished execution");
    }
}
```

```
Run:  Main
/home/shreya/.jdks/openjdk-19.0.2/bin/java -javaagent:/snap/intellij-idea-community/409/lib/idea_rt
thread2 running.
Thread1 running.
thread2 running for 0th time.
thread2 running for 1th time.
Thread1 running for 0th time.
thread2 running for 2th time.
Thread1 running for 1th time.
Thread1 running for 2th time.
thread2 running for 3th time.
Thread1 running for 3th time.
thread2 running for 4th time.
Thread1 running for 4th time.
Thread1 after sleep
thread2 after sleep
Threads have finished execution

Process finished with exit code 0
```

2) Use Synchronize method and synchronize block to enable synchronization between multiple threads trying to access method at same time.

```java
public class Counter {
    private int count=0;
    public synchronized void increment(){
        count++;
    }
    public void decrement() {
        synchronized (this) {
            count--;
        }
    }
    public int getCount(){
        return count;
    }
}
```

```java
public class Mains {
```

```java
    public static void main(String[] args) throws InterruptedException {
        Counter cn=new Counter();
        new Thread( new Runnable() {
            @Override
            public void run() {
                for (int i = 0; i < 10000; i++) {
                    cn.increment();
                }
                System.out.println(cn.getCount());
            }
        }).start();
            new Thread(new Runnable() {
            @Override
            public void run() {
                for (int i = 0; i < 10000; i++) {
                    cn.decrement();
                }
                System.out.println(cn.getCount());
            }
        }).start();
    }
}
```

```
Run:  Mains
/home/shreya/.jdks/openjdk-19.0.2/bin/java -javaagent:/snap/int
100
0

Process finished with exit code 0
```

3) WAP to showcase the usage of volatile in java.

```java
public class VolatileEx {
    public static volatile int count=0;
public static void main(String[] args){
    Thread t1=new Thread(()->{
        int localCount=count;
        while(localCount<5){
            if(localCount!=count){
                localCount=count;
                System.out.println("Count is changed "+localCount);
            }
        }
    });
    Thread t2=new Thread(()->{
        int localcount=count;
        while(localcount<5){
```

```
            System.out.println("Thread 2 increments count to "+localcount);
            count=++localcount;
            try{
                Thread.sleep(100);
            }
            catch (InterruptedException e){
                System.out.println("Exception occurred.");
            }
        }
    });
    t1.start();
    t2.start();
}
}
```

```
VolatileEx
/home/shreya/.jdks/openjdk-19.0.2/bin/java -javaagent:/snap/intellij-idea-community/46
Thread 2 increments count to 0
Count is changed 1
Thread 2 increments count to 1
Count is changed 2
Thread 2 increments count to 2
Count is changed 3
Thread 2 increments count to 3
Count is changed 4
Thread 2 increments count to 4
Count is changed 5

Process finished with exit code 0
```

4) Write a code to simulate a deadlock in java

```
public class BankAccount {
    private float balance;
    private int accountNumber;
    public BankAccount(int accountNumber,float balance){
        this.accountNumber=accountNumber;
        this.balance=balance;
    }
    public synchronized void withdrawal(float amount){
        if(balance<amount){
            System.out.println("Not enough balance");
            return;
        }
        else{
            balance-=amount;
```

```
            System.out.println("Withdrawal of amount "+amount+" is successful.
Balance is "+balance);
        }
    }
    public synchronized void deposit(float amount){
        balance+=amount;
        System.out.println("Balance is "+balance);
    }

}
```

```
public class DeadlockEx {
    public static void main(String[] args){
        BankAccount account1=new BankAccount(1234,344.56F);
        BankAccount account2=new BankAccount(1235,344.56F);
        new Thread("Thread 2") {
            @Override
            public void run() {
                synchronized (account1) {
                    account1.withdrawal(14.5F);
                    try {
                        Thread.sleep(500);
                    } catch (InterruptedException e) {
                        System.out.println("Thread 01 InterruptedException
occurred");
                    }
                    System.out.println("waiting for account 2");

                    synchronized (account2) {
                        account2.deposit(1335.6F);
                    }
                }
            }
        }.start();
        new Thread("thread 1") {
            @Override
            public void run() {
                synchronized (account2) {
                    account2.withdrawal(24.5F);
                    try {
                        Thread.sleep(500);
                    } catch (InterruptedException e) {
                        System.out.println("Thread 02 InterruptedException
occurred");
                    }
                    System.out.println("waiting for account 1");

                    synchronized (account1) {
                        account1.deposit(2345.6F);
```

```
                    }
                }
            }
        }.start();
    }
}
```

```
DeadlockEx
/home/shreya/.jdks/openjdk-19.0.2/bin/java -javaagent:/snap/intellij-idea-community/46
Withdrawal of amount 14.5 is successful. Balance is 330.06
Withdrawal of amount 24.5 is successful. Balance is 320.06
waiting for account 2
waiting for account 1
```