**Experiment No: 5.1**

**Aim:**

|  |
| --- |
| Write a program to demonstrate how threads can be created using Thread class and Runnable interface. |

**Code:**

|  |
| --- |
| **class** A **extends** Thread **implements** Runnable {  **public** **void** run() {  System.***out***.println(Thread.*currentThread*().getName());  }  }  **public** **class** NewClass {  **public** **static** **void** main(String [] as) {  A thrd1 = **new** A();  A thrd2 =**new** A();  thrd1.start();  thrd2.start();  }  } |

**Output:**

|  |
| --- |
| Thread-0  Thread-1 |

**Experiment No: 5.2**

**Aim:**

|  |
| --- |
| Write a program to show difference of calling the start() and run() methods explicitly on thread objects. |

**Code:**

|  |
| --- |
| **package** newclass;  **class** A **implements** Runnable{  **public** **void** run(){  **for**(**int** i=0;i<5;i++){  System.***out***.println("Thread1 called for"+i);  }  }  }  **class** B **extends** Thread{  **public** **void** run(){  **for**(**int** i=0;i<5;i++){  System.***out***.println("Thread2 called for"+i);  }  }  }  **public** **class** NewClass {  **public** **static** **void** main(String[] args) {  System.***out***.println("MAIN START");  A ob=**new** A();  B ob1=**new** B();  Thread t1=**new** Thread(ob);  t1.start();  Thread t2=**new** Thread(ob1);  t2.run();  System.***out***.println("MAIN END");  }    } |

**Output:**

|  |
| --- |
| MAIN START  Thread2 called for0  Thread2 called for1  Thread2 called for2  Thread2 called for3  Thread2 called for4  MAIN END  Thread1 called for0  Thread1 called for1  Thread1 called for2  Thread1 called for3  Thread1 called for4 |

**Experiment No: 5.3**

**Aim:**

|  |
| --- |
| Create two threads T1 and T2. The thread T1 should print numbers from 1 to 10 and thread T2 prints characters from A to J. Ensure that T2 starts first and T1 should only start when T2 finishes. (Note: use join()). |

**Code:**

|  |
| --- |
| **class** A **extends** Thread{  **public** **void** run(){  **for**(**int** i=0;i<10;i++){  System.***out***.println("NUMBER "+i+"\n");  }  }  }  **class** B **extends** Thread{  **public** **void** run(){  **for**(**char** i='A';i<='J';i++){  System.***out***.println("ALPHABET "+i+"\n");  }  }  }  **public** **class** NewClass {  **public** **static** **void** main(String[] args) {  System.***out***.println("MAIN START");  A ob=**new** A();  B ob1=**new** B();  Thread t1=**new** Thread(ob);  Thread t2=**new** Thread(ob1);  t2.start();  **try**  {  System.***out***.println("current THREAD "+Thread.*currentThread*().getName());  t2.join();  }  **catch**(Exception ex)  {  System.***out***.println("Exception"+ ex);  }  t1.start();  System.***out***.println("MAIN END");  }    } |

**Output:**

|  |
| --- |
| MAIN START  current THREAD main  ALPHABET A  ALPHABET B  ALPHABET C  ALPHABET D  ALPHABET E  ALPHABET F  ALPHABET G  ALPHABET H  ALPHABET I  ALPHABET J  MAIN END  NUMBER 0  NUMBER 1  NUMBER 2  NUMBER 3  NUMBER 4  NUMBER 5  NUMBER 6  NUMBER 7  NUMBER 8  NUMBER 9 |

**Experiment No: 5.4**

**Aim:**

|  |
| --- |
| Write a program to show that once an interrupt() method is called upon a thread then even after perfoming some opertions whenever that thread goes to sleep, it would be interrupted. |

**Code:**

|  |
| --- |
| **class** A **extends** Thread{  **public** **void** run(){  Thread t=Thread.*currentThread*();  **for**(**int** i=1;i<=10;i++){  System.***out***.println("Child Thread"+t.getName()+" Is interrupted"+t.isInterrupted());  }  System.***out***.println("Child thread going to sleep");  **try**{  *sleep*(5000);  }  **catch**(Exception e){  System.***out***.println("Child thread interrupted in sleep");    }  System.***out***.println("CHILD THREAD ENDS");  }  }  **public** **class** NewClass **extends** Thread {  **public** **static** **void** main(String[] args){  System.***out***.println("MAIN START");  A ob=**new** A();  Thread t1=**new** Thread(ob,"t1");  t1.start();  System.***out***.println("MAIN END");  t1.interrupt();  }  } |

**Output:**

|  |
| --- |
| MAIN START  MAIN END  Child Threadt1 Is interruptedtrue  Child Threadt1 Is interruptedtrue  Child Threadt1 Is interruptedtrue  Child Threadt1 Is interruptedtrue  Child Threadt1 Is interruptedtrue  Child Threadt1 Is interruptedtrue  Child Threadt1 Is interruptedtrue  Child Threadt1 Is interruptedtrue  Child Threadt1 Is interruptedtrue  Child Threadt1 Is interruptedtrue  Child thread going to sleep  Child thread interrupted in sleep  CHILD THREAD ENDS |

**Experiment No: 5.5**

**Aim:**

|  |
| --- |
| Create a class named Display. Create two methods inside Display. Name one method as dispN() which prints numbers from 1 to 10. Name the second method as dispC() which prints characters from A to J. Create two threads T1 and T2 such that T1 runs the dispN() method and T2 runs the dispC() method. Ensure that when either of thread starts it should complete till end. |

**Code:**

|  |
| --- |
| **class** A **extends** Thread{  **public** **void** disN(){  System.***out***.println("current THREAD "+Thread.*currentThread*().getName());  **for**(**int** i=0;i<10;i++){  System.***out***.println("NUMBER "+i+"\n");  }  }  **public** **void** disC(){  System.***out***.println("current THREAD "+Thread.*currentThread*().getName());  **for**(**char** i='A';i<='J';i++){  System.***out***.println("ALPHABET "+i+"\n");  }  }  }  **class** B **extends** A **implements** Runnable{  A ob;  B(A object){  ob=object;  }  **public** **void** run() {  ob.disN();  }  }  **class** C **extends** A **implements** Runnable{  A ob;  C(A object){  ob=object;  }  **public** **void** run() {  ob.disC();  }  }  **public** **class** NewClass **extends** Thread {  **public** **static** **void** main(String[] args){  System.***out***.println("MAIN START");  System.***out***.println("current THREAD "+Thread.*currentThread*().getName());  A ob=**new** A();  B ob1=**new** B(ob);  C ob2=**new** C(ob);  ob1.start();  **try** {  ob1.join();  } **catch** (InterruptedException e) {  }  ob2.start();  **try** {  *sleep*(3000);  } **catch** (InterruptedException e) {  }  System.***out***.println("MAIN END");  }    } |

**Output:**

|  |
| --- |
| MAIN START  current THREAD main  current THREAD Thread-1  NUMBER 0  NUMBER 1  NUMBER 2  NUMBER 3  NUMBER 4  NUMBER 5  NUMBER 6  NUMBER 7  NUMBER 8  NUMBER 9  current THREAD Thread-2  ALPHABET A  ALPHABET B  ALPHABET C  ALPHABET D  ALPHABET E  ALPHABET F  ALPHABET G  ALPHABET H  ALPHABET I  ALPHABET J  MAIN END |

**Experiment No: 5.6**

**Aim:**

|  |
| --- |
| Write a program to build an application where two threads named "T1" and "T2" exist. T1 calculates the sum of N integers and T2 wants to use the sum generated by T1 to calculate the average. Use join() to synchronize their activities. |

### Code:

|  |
| --- |
| **class** A{  **int** sum=0;  **int** n;  **int** avg;  **public** **void** getNSum(**int** q){  n=q;  System.***out***.println("current THREAD "+Thread.*currentThread*().getName());  **for**(**int** i=1;i<=n;i++) {  sum=sum+i;  }  System.***out***.println("SUM IS " +sum);  }  **public** **void** getNAvg() {  System.***out***.println("current THREAD "+Thread.*currentThread*().getName());  avg=sum/n;  System.***out***.println("AVERAGE IS "+avg);  }  }  **class** thread1 **extends** Thread{  A ob;  thread1(A object){  ob=object;  }  **public** **void** run() {  ob.getNSum(111);  }  }  **class** thread2 **extends** Thread{  A ob;  thread2(A object){  ob=object;  }  **public** **void** run() {  ob.getNAvg();  }  }  **public** **class** NewClass **extends** Thread {  **public** **static** **void** main(String [] as) {  A ob=**new** A();  thread1 t1=**new** thread1(ob);  thread2 t2=**new** thread2(ob);  t1.start();  **try** {  t1.join();  } **catch** (InterruptedException e) {  }  t2.start();  }  } |

**Output:**

|  |
| --- |
| current THREAD Thread-0  SUM IS 6216  current THREAD Thread-1  AVERAGE IS 56 |

**Experiment No: 5.7**

**Aim:**

|  |
| --- |
| Demonstrate using a Java program, how DEADLOCK occurs between threads and also give solution program. |

**Code:**

|  |
| --- |
| **class** T1 **extends** Thread {  **static** Thread *waitFor*;  **public** **void** run() {  System.***out***.println("T1 waiting for T2");  **try**{*waitFor*.join();}**catch**(Exception e){}  **for** (**int** i = 1; i<=3; i++) {  System.***out***.println("T1 executing");  **try**{Thread.*sleep*(1000);}**catch**(Exception e){}  }  }  }  **class** T2 **extends** Thread {  **static** Thread *waitFor*;  **public** **void** run() {  System.***out***.println("T2 waiting for T1");  **try**{*waitFor*.join();}**catch**(Exception e){}  **for** (**int** i = 1; i <= 3; i++) {  System.***out***.println("T2 executing");  **try**{Thread.*sleep*(1000);}**catch**(Exception e){}  }  }  }  **public** **class** Test {  **public** **static** **void** main(String[] arrr) {  T1 t1=**new** T1();  T2 t2=**new** T2();  T2.*waitFor*=t1;  T1.*waitFor*=t2;  t1.start();  t2.start();  }  } |
| **Output:**  T1 waiting for T2  T2 waiting for T1 |

**Experiment No: 5.8**

**Aim:**

|  |
| --- |
| Write a program to build an application where two threads named "T1" and "T2" can withdraw money from a single account.Following are the rules: Initially the balance should be 2000.One thread can withdraw a minimum amount of 1000 but the balance should not go below 1000.At every successful withdrawal the balance amount should be displayed.The withdrawal process should be repeated atleast 4 times by "T1" and 4 times by "T2".(Note: use synchronized method or block). |

**Code:**

|  |
| --- |
| **class** Account **extends** Thread{  **private** **int** bal=2000;  **public** **synchronized** **void** withdraw(**int** wtd) {  System.***out***.println("current THREAD @ "+Thread.*currentThread*().getName());  **if** (wtd>1000) {  System.***out***.println("MINIMUM AMOUNT WITHRAWAL IS 1000");  }  **else** **if**(bal<=1000) {  System.***out***.println("INSUFFICANT BALANCE");  }  **else** {  bal=bal-wtd;  **if**(bal<=1000) {  System.***out***.println("INSUFIICAINT BALANCE ");  bal=bal+wtd;  }  }  }  **public** **void** ShowBal() {  System.***out***.println("THE CURRENT BALANCE IS "+bal);  }  }  **class** person1 **extends** Thread{  Account a;  person1(Account ob1){  a=ob1;  }  **public** **void** run() {  **for**(**int** i=0;i<5;i++) {  a.withdraw(100);  a.ShowBal();  }  }}  **class** person2 **extends** Thread{  Account a;  person2(Account ob2){  a=ob2;  }  **public** **void** run() {  **for**(**int** i=0;i<5;i++) {  a.withdraw(600);  a.ShowBal();  } }  }  **public** **class** NewClass **extends** Thread {  **public** **static** **void** main(String as []) {  System.***out***.println("Main Start");  System.***out***.println("current THREAD "+Thread.*currentThread*().getName());  Account a1=**new** Account();  person1 ob1=**new** person1(a1);  person2 ob2=**new** person2(a1);  ob1.start();  ob2.start();  System.***out***.println("Main end");  }  } |

**Output:**

|  |
| --- |
| Main Start  current THREAD main  Main end  current THREAD @ Thread-1  THE CURRENT BALANCE IS 1900  current THREAD @ Thread-2  THE CURRENT BALANCE IS 1300  current THREAD @ Thread-1  THE CURRENT BALANCE IS 1200  current THREAD @ Thread-1  THE CURRENT BALANCE IS 1100  current THREAD @ Thread-1  INSUFIICAINT BALANCE  THE CURRENT BALANCE IS 1100  current THREAD @ Thread-1  INSUFIICAINT BALANCE  THE CURRENT BALANCE IS 1100  current THREAD @ Thread-2  INSUFIICAINT BALANCE  THE CURRENT BALANCE IS 1100  current THREAD @ Thread-2  INSUFIICAINT BALANCE  THE CURRENT BALANCE IS 1100  current THREAD @ Thread-2  INSUFIICAINT BALANCE  THE CURRENT BALANCE IS 1100  current THREAD @ Thread-2  INSUFIICAINT BALANCE  THE CURRENT BALANCE IS 1100 |

**Experiment No: 5.9**

**Aim:**

|  |
| --- |
| Demonstrate using a Java program, how STARVATION occurs between threads and also give solution program. |

**Code:**

|  |
| --- |
| **class** A **extends** Thread{  **public** **void** run() {  **while**(**true**) {  System.***out***.println("THREAD A");  }  }  }  **class** B **extends** Thread{  **public** **void** run() {  System.***out***.println("THREAD B");  }  }  **public** **class** NewClass **extends** Thread{  **public** **static** **void** main(String as[]) {  A ob1=**new** A();  B ob2=**new** B();  ob1.run();  **try** {  ob1.join();  } **catch** (InterruptedException e) {  // **TODO** Auto-generated catch block  }  ob2.run();  }  } |

**Output:**

|  |
| --- |
| THREAD A  THREAD A.  .  .  .  .  THREAD A WILL RUN INFINITY.  THREAD B WILL STARVE. |