Lab-6

June 23, 2021

1) Create a child thread to display the message 10 times “Child Thread” and “Parent Thread” using thread class

```
class child extends Thread
{
        public void run()
        {
        for(int i=0;i<10;i++)
        {
            System.out.println(i+" child Thread");
        }
}
}
```

```
public class childThread
{
public static void main(String args[])
{
```

```
child c = new child();
```

```
        c.start();
        try{
        c.join();
        }
        catch(Exception e)
        {
        System.out.println(e);
        }
```

```
        for(int i=0;i<10;i++)
        {
        System.out.println(i+" parent Thread");
}
}
}
```

Output:

```
0 child Thread
1 child Thread
2 child Thread
3 child Thread
4 child Thread
5 child Thread
6 child Thread
7 child Thread
8 child Thread
9 child Thread
0 parent Thread
1 parent Thread
2 parent Thread
3 parent Thread
4 parent Thread
5 parent Thread
6 parent Thread
7 parent Thread
8 parent Thread
9 parent Thread
PS C:\Users\Asus\Desktop\GIT>
```

2) Create a child thread to display the message 10 times “Child Thread” and “Parent Thread” using runnable interface

```
class child2 implements Runnable
{
    public void run()
        {
        for(int i=0;i<10;i++)
        {
            System.out.println(i+" child Thread");
        }
        }
}
```

```
public class childThread
{
public static void main(String args[])
{
```

```
child2 t = new child2();
Thread c = new Thread(t);
```

```
        c.start();
        try{
        c.join();
        }
        catch(Exception e)
        {
        System.out.println(e);
        }
```

```
        for(int i=0;i<10;i++)
        {
        System.out.println(i+" parent Thread");
}
}
}
```

Output:

```
0 child Thread
1 child Thread
2 child Thread
3 child Thread
4 child Thread
5 child Thread
6 child Thread
7 child Thread
8 child Thread
9 child Thread
0 parent Thread
1 parent Thread
2 parent Thread
3 parent Thread
4 parent Thread
5 parent Thread
6 parent Thread
7 parent Thread
8 parent Thread
9 parent Thread
PS C:\Users\Asus\Desktop\GIT>
```

3) Develop a program to create three child threads that performs the following operations i) The first thread displays numbers which are multiples of 10 from 1 to 100. ii) The second thread displays all even numbers from 1 to 100. iii) The third thread displays numbers
which are multiples of 5 between 1 and100.

4) For the above program, by calling the appropriate methods ensure that the main thread is the last to execute and display whether the threads are alive. Hint : use isalive() and join().

```
class multen extends Thread
{
    public void run()
    {
        for(int i=1;i<=100;i++)
        {
            if(i%10==0)
                System.out.println("Multiple of 10 - "+i);
        }
    }
}

class even extends Thread
{
    public void run()
    {
        for(int i=1;i<=100;i++)
        {
            if(i%2==0)
             System.out.println("even number - "+i);
        }
    }
}
```

```
class mulfive extends Thread
{
    public void run()
    {
        for(int i=1;i<=100;i++)
        {
            if(i%5==0)
                System.out.println("Multiple of 5 - "+i);
        }
    }
}
```

```
public class threeThread
{
public static void main(String args[])
    {
    multen obj1 = new multen();
    even obj2 = new even();
    mulfive obj3 = new mulfive();
```

```
    obj1.start();
    obj2.start();
    obj3.start();
```

```
    System.out.println("Thread 1 IS "+obj1.isAlive());
    System.out.println("Thread 2 IS "+obj2.isAlive());
    System.out.println("Thread 3 IS "+obj3.isAlive());
```

```
    try{
        obj1.join();
        obj2.join();
        obj3.join();
    }
    catch(Exception e)
    {
        System.out.println(e);
    }
    System.out.println("main is the last to execute");
```

```
    }
}
```

Output:

```
even number - 40
Multiple of 10 - 100
even number - 42
Multiple of 5 - 50
Multiple of 5 - 55
Multiple of 5 - 60
even number - 44
Multiple of 5 - 65
even number - 46
Multiple of 5 - 70
even number - 48
Multiple of 5 - 75
even number - 50
even number - 52
Multiple of 5 - 80
even number - 54
Multiple of 5 - 85
Multiple of 5 - 90
even number - 56
Multiple of 5 - 95
even number - 58
Multiple of 5 - 100
even number - 60
even number - 62
even number - 64
even number - 66
even number - 68
even number - 70
even number - 72
even number - 74
even number - 76
even number - 78
even number - 80
even number - 82
even number - 84
even number - 86
even number - 88
even number - 90
even number - 92
even number - 94
even number - 96
even number - 98
even number - 100
main is the last to execute
```

5) Write a program to implement a dynamic growable queue using generics.

Code:

```
import java.util.Scanner;
import java.util.ArrayList;
    class queue<T>
    {
    ArrayList<T> q=new ArrayList<T>();
    int currentSize=0;
    void enqueue(T a)
    {
        q.add(a);
        currentSize++;
    }
    void dequeue()
    {
        if(q.isEmpty()==true)
        {
            System.out.println("queue is empty cant dequeue");
        }
        else
        {
            System.out.println(q.get(0)+" is dequed");
            q.remove(0);
            currentSize--;
        }
    }
}
class QueueGrowable
{
    public static void main(String[] args)
    {
        queue<Integer> S=new queue<Integer>();
        queue<Double> S2=new queue<Double>();
        queue<String> S3=new queue<String>();
        Scanner sc=new Scanner(System.in);
        int x,y;
        System.out.println("enter 1 for integer");
        System.out.println("enter 2 for double");
        System.out.println("enter 3 for string");
        y=sc.nextInt();
        if(y==1){
        do
        {
            System.out.println("enter 0 to exit");
            System.out.println("enter 1 to enqueue");
            System.out.println("enter 2 to dequeue");
            // System.out.println("enter 3 to display top element");
            x=sc.nextInt();

            switch(x)
            {
                case 1 : int n;
                System.out.println("enter element to be pushed:");
                n=sc.nextInt();
                S.enqueue(n);
```

```
                break;
                case 2 : S.dequeue();
                break;
                // case 3 : S2.displayTop();
                // break;
                case 0 : break;
            }
        } while(x>0);
    }
        if(y==2){
        do
        {
            System.out.println("enter 0 to exit");
            System.out.println("enter 1 to enqueue");
            System.out.println("enter 2 to dequeue");
            // System.out.println("enter 3 to display top element");
            x=sc.nextInt();

            switch(x)
            {
                case 1 : double n;
                System.out.println("enter element to be pushed:");
                n=sc.nextDouble();
                S2.enqueue(n);
                break;
                case 2 : S2.dequeue();
                break;
                // case 3 : S2.displayTop();
                // break;
                case 0 : break;
            }
        } while(x>0);
    }
        if(y==3){
        do
        {
            System.out.println("enter 0 to exit");
            System.out.println("enter 1 to enqueue");
            System.out.println("enter 2 to dequeue");
            // System.out.println("enter 3 to display top element");
            x=sc.nextInt();

            switch(x)
            {
                case 1 :
                String n;
                System.out.println("enter element to enqueue:");
                n=sc.next();
                S3.enqueue(n);
                break;
                case 2 : S3.dequeue();
                break;
                // case 3 : S3.displayTop();
                // break;
                case 0 : break;
            }
        } while(x>0);
    }

    }
    }
```

Output:

```
enter 1 for integer
enter 2 for double
enter 3 for string
1
enter 0 to exit
enter 1 to enqueue
enter 2 to dequeue
1
enter element to be pushed:
12
enter 0 to exit
enter 1 to enqueue
enter 2 to dequeue
2
12 is dequed
enter 0 to exit
enter 1 to enqueue
enter 2 to dequeue
```

6)Consider a Bus reservation system that allows online reservations to its customers. Suppose there are two transactions of reservation for a particular seat simultaneously which leads to race condition. Develop a solution to avoid the unpredictable situation with a program.

```
class TicketBooking implements Runnable{
    int ticketsAvailable=1;
      public void run(){
           System.out.println("Waiting to book ticket for : "+Thread.currentThread().getName())
;
           synchronized (this) {
                 if(ticketsAvailable>0){
                       System.out.println("Booking ticket for :"+Thread.currentThread().getNa
me());

                       try{
                             Thread.sleep(1000);
                       }catch(Exception e){}
                       ticketsAvailable--;
                       System.out.println("Ticket BOOKED for : "+  Thread.currentThread().get
Name());
                       System.out.println("currently ticketsAvailable = "+ticketsAvailable);
                 }
                 else{
```

```
                    System.out.println("Ticket NOT BOOKED for : "+
                                Thread.currentThread().getName());
                }
        }
}
 }
 public class busReservation {
   public static void main(String args[])
   {
          TicketBooking obj=new TicketBooking();

          Thread thread1=new Thread(obj,"Passenger 1 Thread");
          Thread thread2=new Thread(obj,"Passenger 2 Thread");

          thread1.start();
          thread2.start();
   }
}
```

Output:

```
Waiting to book ticket for : Passenger 1 Thread
Booking ticket for :Passenger 1 Thread
Waiting to book ticket for : Passenger 2 Thread
Ticket BOOKED for : Passenger 1 Thread
currently ticketsAvailable = 0
Ticket NOT BOOKED for : Passenger 2 Thread
PS C:\Users\Asus\Desktop\GIT>
```