Assignment 1:

Write a java program which accepts multiple employees details,

1. **Create thread class**
2. **Execute them using frokjoinpool**
3. **make the use of runnable interface in it.**

Source Code: *(Black colour in background is due to dark mode of IDE)*

Employe.java

```java
public class Employe {
    private int empid;
    private double empsalary;
    private String empname;

    public String getName() {
        return empname;
    }

    public void setName(String name) {
        this.empname = name;
    }

    public double getSalary() {
        return empsalary;
    }

    public void setSalary(double salary) {
        this.empsalary = salary;
    }

    public int getId() {
        return empid;
    }

    public void setId(int id) {
        this.empid = id;
    }
}
```

Emoloyegen.java

```java
import java.util.*;

public class Employegen {
    public List<Employe> generate(int size) {
        List<Employe> emp = new ArrayList<Employe>();
        for (int i = 0; i < size; i++) {
            Employe employe = new Employe();
            employe.setName("emp" + (i + 1));
            employe.setId(i + 1);
            employe.setSalary(1000.0);
            emp.add(employe);
        }
        return emp;
    }
}
```

Thread.java

```java
import java.util.*;
import java.util.concurrent.RecursiveAction;

public class Thread extends RecursiveAction {
    private List<Employe> employes;
    private int first;
    private int last;
    private double increment;

    public Thread(List<Employe> Employes, int first, int last, double increment) {
        this.employes = Employes;
        this.first = first;
        this.last = last;
        this.increment = increment;
    }

    protected void compute() {
        if (last - first < 10) {
            updateSalary();
        } else {
            int middle = (first + last) / 2;
            System.out.printf("Task pending tasks: %s\n", getQueuedTaskCount());
            Thread t1 = new Thread(employes, first, middle + 1, increment);
            Thread t2 = new Thread(employes, middle + 1, last, increment);
            invokeAll(t1, t2);

        }
    }
    private void updateSalary() {
        for (int i = first; i < last; i++) {
            Employe employe = employes.get(i);
            employe.setSalary((employe.getSalary()) * 2);
        }
    }
}
```

Main.java

```java
import java.util.List;
import java.util.concurrent.ForkJoinPool;
import java.util.concurrent.TimeUnit;


class Main{
    public static void main(String args[]) {
    Employeegen gen= new Employeegen();
    List<Employe> employes= gen.generate(10);
    Thread thread=new Thread(employes,0,employes.size(),0.20);
    for(int i=0;i<employes.size();i++) {
        Employe employ=employes.get(i);
        System.out.printf("Employe %s: %f \n",employ.getName(),employ.getSalary());
    }
    System.out.println("------------------------------------------------------------------
------------------------");
    System.out.println("To Increase the salary of Employes");
    System.out.println("------------------------------------------------------------------
------------------------");
    ForkJoinPool pool=new ForkJoinPool();
    pool.execute(thread);
    do {
        System.out.printf("**************************************\n");
        System.out.printf("Main: Pralleism:%d\n", pool.getCommonPoolParallelism());
    }while(!thread.isDone());
    pool.shutdown();

    if(thread.isCompletedNormally()) {
        System.out.println("Main: The process has completed normally. \n");
    }
    for(int i=0;i<employes.size();i++) {
        Employe employ=employes.get(i);
        System.out.printf("Employe %s: %f \n",employ.getName(),employ.getSalary());
    }
    }
}
```

Output:

```
Employe emp1: 1000.000000
Employe emp2: 1000.000000
Employe emp3: 1000.000000
Employe emp4: 1000.000000
Employe emp5: 1000.000000
Employe emp6: 1000.000000
Employe emp7: 1000.000000
Employe emp8: 1000.000000
Employe emp9: 1000.000000
Employe emp10: 1000.000000
-------------------------------------------------------------------------------------------
To Increase the salary of Employes
-------------------------------------------------------------------------------------------
**************************************
Main: Pralleism:7
**************************************
Main: Pralleism:7
**************************************
Main: Pralleism:7
**************************************
Main: Pralleism:7
**************************************
Main: Pralleism:7
**************************************
Main: Pralleism:7
**************************************
Task pending tasks: 0
Main: Pralleism:7
**************************************
Main: Pralleism:7
**************************************
Main: Pralleism:7
**************************************
Main: Pralleism:7
**************************************
Main: Pralleism:7
Main: The process has completed normally.

Employe emp1: 2000.000000
Employe emp2: 2000.000000
Employe emp3: 2000.000000
Employe emp4: 2000.000000
Employe emp5: 2000.000000
Employe emp6: 2000.000000
Employe emp7: 2000.000000
Employe emp8: 2000.000000
Employe emp9: 2000.000000
Employe emp10: 2000.000000
```