```
CECS 277
LAB ASSIGNMENT 4
Due date: Wednesday 9/30
Name: Tina L Vu
EmployeeInfo Class:
public interface EmployeeInfo {
  //constant values
  double FACULTY_MONTHLY_SALARY = 5000.00;
  int STAFF_MONTHLY_HOURS_WORKED = 160;
}
Employee Class:
import java.util.Comparator;
public abstract class Employee implements Comparator, Comparable {
  //instance variables
  String last name;
  String first name;
  String ID_number;
  /**default argument constructor
   * given no param
  */
  Employee(){
    String last_name = "";
    String first_name = "";
    String ID_number = "";
  }
  /**override argument constructors
    @param last name: the last name of the employee (String)
   * @param first name: the first name of the employee (String)
   * @param ID_number: the ID number of the employee (String)
  Employee(String last_name, String first_name, String ID_number){
    setLast_name(last_name);
    setFirst name(first name);
    setID_number(ID_number);
  }
  //public methods (include mutators and accessors)
   * @Overrided toString
   * @return a string of the employee's id number and full name
  public String toString() {
```

```
return "ID Employee number: " + ID number +
       "\nEmployee Name: " + first_name + " " +
       last_name;
}
//set first name of employee
public void setFirst_name(String first_name) {
  this.first_name = first_name;
//set last name of employee
public void setLast_name(String last_name) {
  this.last_name = last_name;
//set id of employee
public void setID_number(String ID_number) {
  this.ID_number = ID_number;
}
//get first name of employee
public String getFirst_name() {
  return first name;
}
//get last name of employee
public String getLast_name() {
  return last_name;
//get id of employee
public String getID_number() {
  return ID_number;
//body in child methods will return monthly wage
abstract public double monthlyEarning();
//use to sort id in ascending order using comparable
public int compareTo(Employee id){
  return Integer.parseInt(ID_number) - Integer.parseInt(id.getID_number());
}
//use to sort last name in descending order using comparator
public static int compare(Employee last1, Employee last2){
  return last1.getLast_name().compareTo(last2.getLast_name());
}
```

}

Staff Class:

```
//Staff class extending from class Employee
public class Staff extends Employee implements EmployeeInfo{
  //instance variables
  double hourly_rate;
  /**default argument constructor
   * given no param
   */
  Staff() {
    super();
    hourly_rate = 0;
  }
  /**override argument constructors
   * @param last name : last name of employee
   * @param first_name : first name of employee
   * @param ID_number : id of employee
   * @param hourly_rate : hourly rate of employee
  Staff(String last_name, String first_name, String ID_number, double hourly_rate) {
     super(last_name, first_name, ID_number);
    setHourly_rate(hourly_rate);
  }
  //methods
  /** accessor
   * get the employee's hourly rate
   * @return the hourly rate
  public double getHourly_rate() {
     return hourly_rate;
  }
   * get the monthly earning in a string format
   * @return monthly earning
  public String getMonthlyEarning(){
     return String.format("%.2f", monthlyEarning());
  /** mutator
   * set hourly rate
   * @param hourly_rate: the hourly rate
  public void setHourly_rate(double hourly_rate) {
```

```
this.hourly_rate = hourly_rate;
  }
  /**@Override monthlyEarning
   * @return the employee's monthly salary
  public double monthlyEarning() {
    return hourly_rate * STAFF_MONTHLY_HOURS_WORKED;
  /**@Override toString
  * @return the employee's id, full name, and monthly salary
   */
  public String toString() {
    return super.toString() + "\nFull time \n" + "Monthly Salary: $" + getMonthlyEarning();
  }
  @Override
  public int compare(Object o1, Object o2) {
    return 0;
}
```

```
Education Class:
public class Education {
  //instance variables
  private String Degree;
  private String Major;
  private int Research;
  /**default argument constructor
   * given no param
   */
  Education() {
    Degree = "";
    Major = "";
    Research = 0;
  /**override argument constructors
    @param degree: the degree of the employee (MS or PhD)
   * @param major: the major they took (EX. Engineering, English, etc.)
   * @param research: number of researches made
   */
  Education(String degree, String major, int research) {
    setDegree(degree);
    setMajor(major);
    setResearch(research);
  }
  //accessors
   * get the degree of the employee
   * @return that degree as a string; either MS or PhD
   */
  public String getDegree() {
    return Degree;
  }
   * get the major of that employee
   * @return the major as a string
  public String getMajor() {
    return Major;
  }
   * get the number of researches that employee had made
   * @return an integer that represent the number of researches
```

```
public int getResearch() {
     return Research;
  }
  //settors
  //set the degree of the employee with the given param
  public void setDegree(String degree) {
     Degree = degree;
  }
  //set the major of the employee with the given param
  public void setMajor(String major) {
     Major = major;
  }
  //set the researched number w/the given param
  public void setResearch(int research) {
     Research = research;
  }
}
```

Faculty Class:

public class Faculty extends Employee implements EmployeeInfo{

```
//constants
  public enum Levels{
    AS, AO, FU, NA
  private Education education;
  private Levels IvlEducation;
  /**default argument constructor
   * given no param
   */
  public Faculty(){
    super();
    this.lvlEducation = Levels.NA;
    this.education = new Education();
  }
  /**override argument constructors
   * @param last_name: employee's last name
   * @param first_name: employee's first name
   * @param ID number: employee's id number
   * @param IvlEducation: employee's level of education
  public Faculty(String last_name, String first_name, String ID_number, Levels IvlEducation, String
degree, String major, int research){
    super(last_name, first_name, ID_number);
     setIvIEducation(IvIEducation);
     this.education = new Education(degree, major, research);
  }
  //methods
  //accessors
  public Levels getIvIEducation() {
     return IvlEducation;
  }
  //mutators
  public void setlvlEducation(Levels edu) {
    IvIEducation = edu;
  /**@Override monthlyEarning
   * @return that faculty monthly salary * level of education
  public double monthlyEarning() {
     if(IvlEducation == Levels.AS){
```

```
return FACULTY MONTHLY SALARY;
  } else if (IvlEducation == Levels.AO){
     return FACULTY_MONTHLY_SALARY * 1.5;
  } else if (IvIEducation == Levels.FU){
     return FACULTY_MONTHLY_SALARY * 2.0;
  }
  return 0;
}
/**@Override toString
* @return a String w/ID, Full name, IvI of Education, Degree, Major, & num of research
public String getString() {
  return "Level: " + IvlEducation +
       "\nDegree: " + education.getDegree() + "\nMajor: " + education.getMajor() +
       "\nResearches: " + education.getResearch();
}
/**
* get the monthly earning in a string format
* @return monthly earning
public String getMonthlyEarning(){
  return String.format("%.2f", monthlyEarning());
}
@Override
public String toString(){
  switch (IvIEducation){
     case AS:
       return super.toString() + "\nLevel: ASSISTANT" + "\nMonthly Salary: $" +
            getMonthlyEarning();
       return super.toString() + "\nLevel: ASSOCIATE" + "\nMonthly Salary: $" +
            getMonthlyEarning();
       return super.toString() + "\nLevel: FULL" + "\nMonthly Salary: $" +
            getMonthlyEarning();
  }
  return IvIEducation + "\n";
}
@Override
public int compare(Object o1, Object o2) {
  return 0;
}
```

}

Part-Time Class:

```
public class Partime extends Staff{
  //instance variables
  private int hours_work_per_week;
  /**default argument constructor
  * given no param
   */
  public Partime(){
    super();
    this.hours_work_per_week = 0;
  /**override argument constructors
   * @param last name: employee's last name
   * @param first name: employee's first name
   * @param ID_number: employee's ID number
   * @param hourly rate: the employee's hourly rate
   * @param hours_work_per_week: the employee's total hours of work per week
  public Partime(String last_name, String first_name, String ID_number, double hourly_rate, int
hours_work_per_week) {
    super(last name, first name, ID number, hourly rate);
    setHours_work_per_week(hours_work_per_week);
  }
  //methods
  /**accessors: get the employee's total hours of work per week
   * @return hours in int
  public int getHours work per week() {
    return hours_work_per_week;
  /**mutators: set the employee's total hours of work per week
   * @param hours work per week: the int used to set it
  public void setHours_work_per_week(int hours_work_per_week) {
    this.hours_work_per_week = hours_work_per_week;
  }
  /**@Override monthlyEarning
   * @return monthly salary: the hourly rate times the hours worked in 4 weeks
  public double monthlyEarning() {
    return hourly_rate*(hours_work_per_week*4);
  }
```

```
/**@Override toString

* @return Employee's ID, Name, hours worked per month, and monthly salary

*/

public String toString() {

return "ID Employee Number: " + ID_number + "\nEmployee Name: " +

first_name + " " + last_name + "\nHours Works Per Month: " +

hours_work_per_week + "\nMonthly Salary: " + monthlyEarning();

}
```

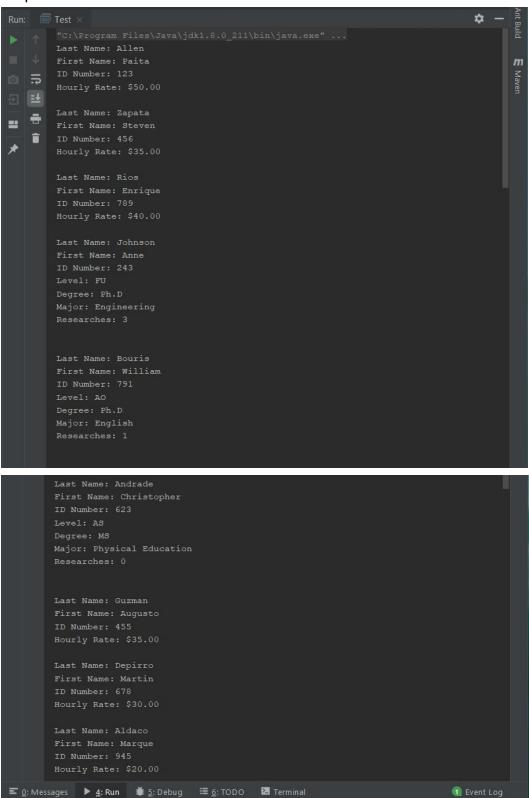
Test Class:

```
import java.util.ArrayList;
import java.util.Collections;
public class Test {
  public static void main(String[] args){
     //time the entire program
     long start = System.nanoTime();
    //create an arraylist of class Employee
     ArrayList<Employee> Employ = new ArrayList<Employee>();
    //store Staff
     Employ.add(new Staff("Allen", "Paita", "123", 50.00));
     Employ.add(new Staff("Zapata", "Steven", "456", 35.00));
     Employ.add(new Staff("Rios", "Enrique", "789", 40.00));
    //store Faculty
     Employ.add(new Faculty("Johnson", "Anne", "243", Faculty.Levels.FU, "Ph.D", "Engineering", 3));
     Employ.add(new Faculty("Bouris", "William", "791", Faculty.Levels.AO, "Ph.D", "English", 1));
     Employ.add(new Faculty("Andrade", "Christopher", "623", Faculty.Levels.AS, "MS", "Physical
Education", 0));
     //store Partime
     Employ.add(new Partime("Guzman", "Augusto", "455", 35.00, 30));
     Employ.add(new Partime("Depirro", "Martin", "678", 30.00, 15));
     Employ.add(new Partime("Aldaco", "Marque", "945", 20.00, 35));
     //my neat output
     for(int j = 0; j < Employ.size(); j++){
       System.out.println("Last Name: " + Employ.get(j).getLast_name());
       System.out.println("First Name: " + Employ.get(j).getFirst_name());
       System.out.println("ID Number: " + Employ.get(j).getID_number());
       if(Employ.get(j) instanceof Staff ){
          System.out.printf("Hourly Rate: $%.2f", ((Staff) Employ.get(j)).getHourly rate());
       else if(Employ.get(j) instanceof Partime ){
          System.out.printf("Hourly Rate: $$%.2f", ((Partime) Employ.get(j)).getHourly_rate());
          System.out.println("Hrs Worked Per Week: " + ((Partime)
Employ.get(j)).getHours work per week());
       } else if(Employ.get(j) instanceof Faculty){
          System.out.println("" + ((Faculty) Employ.get(j)).getString());
       } else{
          System.out.println("Done");
       System.out.println("\n");
    }
```

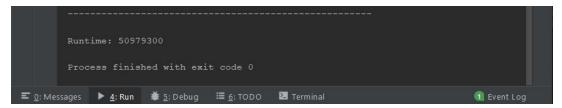
```
System.out.println("-----
     //a)display employee's information using the method toString
     System.out.println("a)display employee's information using the method toString\n");
     for(int j = 0; j < Employ.size(); j++){
       System.out.println(Employ.get(j).toString());
       System.out.println("\n");
    }
    //to make it neat
     System.out.println("-----\n");
     //b & c) finding total monthly salary of part time staffs and all staffs
     System.out.println("b & c) finding total monthly salary of part time staffs and all staffs");
     double total Monthly PartTime Salary = 0.00;
     double total_Monthly_Salary = 0.00;
     //adding it all up
     for(int i = 0; i < Employ.size(); i++){}
       //add total monthly salary for all the part-timer staff
       if(Employ.get(i) instanceof Partime){
         total_Monthly_PartTime_Salary += Employ.get(i).monthlyEarning();
       //add total monthly salary for all employees
       total_Monthly_Salary += Employ.get(i).monthlyEarning();
    }
     //printing out answer for question b & c
     System.out.printf("\nTotal Monthly Salary for All Part-time Staff:
$%.2f",total Monthly PartTime Salary);
     System.out.printf("\nTotal Monthly Salary for All Employee: $%.2f", total_Monthly_Salary);
     //to make it neat
     System.out.println("\n");
     System.out.println("\n----\n");
     //d) Sort id in ascending order using Comparable
     System.out.println("d) Sort id in ascending order using Comparable");
     //sort by id w/out using sort (compareTo)
     Collections.sort(Employ,Employee::compareTo);
     //print
    for(int j = 0; j < Employ.size(); j++){
       System.out.println(Employ.get(j).toString());
       System.out.println("\n");
    }
     //to make it neat
```

```
System.out.println("\n");
    System.out.println("\n-----\n");
    //e) Sort last name in descending order using Comparator
    System.out.println("e) Sort last name in descending order using Comparator");
    //sort by first character of last name w/out using sort (compare)
    Collections.sort(Employ,Employee::compare);
    //print
    for(int j = 0; j < Employ.size(); j++){
       System.out.println(Employ.get(j).toString());
       System.out.println("\n");
    }
    //calculate runtime
    long end = System.nanoTime();
    long time = end - start;
    //runtime output
    System.out.println("Runtime: " + time);
  }
}
```

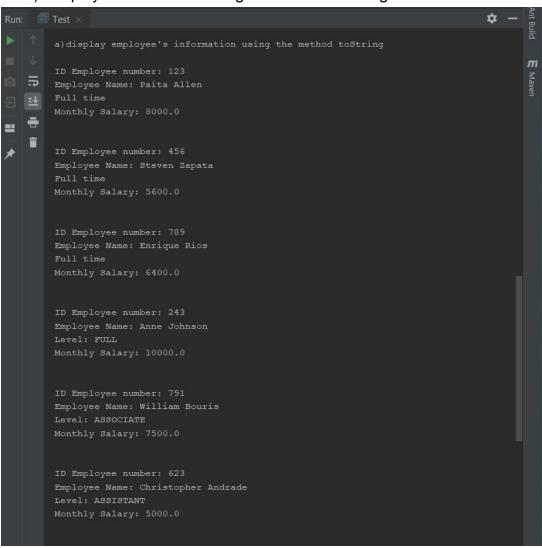
Output:

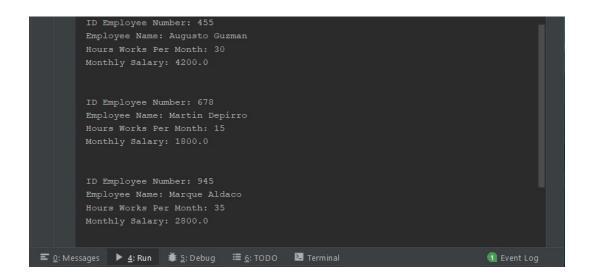


Runtime:



A) Employee information using the method toString





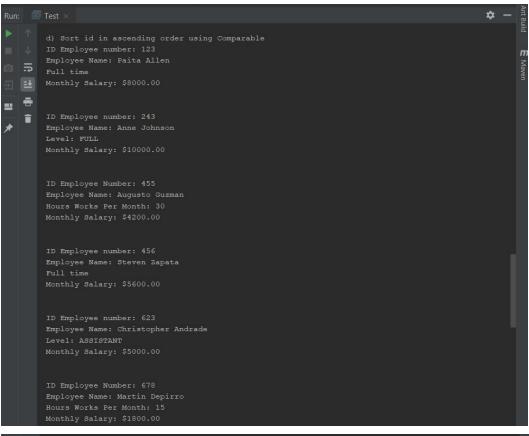
B + C) finding total monthly salary of part time staffs and all staffs

```
b & c) finding total monthly salary of part time staffs and all staffs

Total Monthly Salary for All Part-time Staff: $8800.00

Total Monthly Salary for All Employee: $51300.00
```

D) Sort id in ascending order using Comparable



```
ID Employee number: 789
Employee Name: Enrique Rios
Full time
Monthly Salary: $6400.00

ID Employee number: 791
Employee Name: William Bouris
Level: ASSOCIATE
Monthly Salary: $7500.00

ID Employee Number: 945
Employee Name: Marque Aldaco
Hours Works Per Month: 35
Monthly Salary: $2800.00
```

E) Sort last name in descending order using Comparator

