

Answers for Lab 4
Week 5

1. An educational institution wishes to maintain a database of its employees. The database is divided into a number of classes whose hierarchical relationships are shown in Fig.1. The figure also shows the minimum information required for each class. Specify all the classes and define methods to create the database and retrieve individual information as and when required.

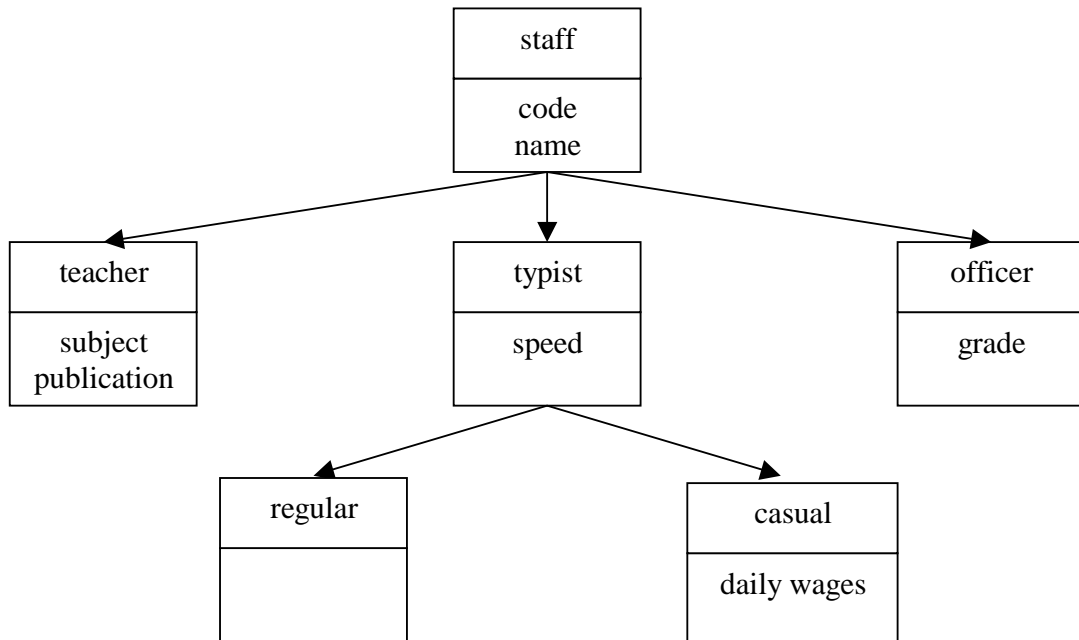


Fig. 1

Sample Answer:

```
/* This class represents all the staff */
class staff
{
    String code;
    String name;

    /* Input parameters for the constructor are:
    String c: staff code.
    String n: staff name.
    */
}
```

```

public staff(String c, String n)
{
    code = c;
    name = n;
}

/* set/get methods for the staff code. */
public void setCode(String c)
{
    code = c;
}
public String getCode()
{
    return code;
}

/* set/get methods for the staff name. */
public void setName(String n)
{
    name = n;
}
public String getName()
{
    return name;
}
}

/* This class represents all the teacher staff */
class teacher extends staff
{
    private String subject;
    private String publication;
    String code;

    /* Input parameters for the constructor are:
       String c: staff code.
       String n: staff name.
    */
    public teacher(String c, String n)
    {
        super(c, n);
    }

    /* Input parameters for the constructor are:
       String c: staff code.
       String n: staff name.
       String sub: teacher subject.
       String pub: array of teacher publication
    */
    public teacher(String c, String n, String sub, String pub)
    {
        super(c, n);
        subject = sub;
        publication = pub;
    }
}

```

```

public void setCode(String s)
{
    super.setCode(s);
}

/* set/get methods for subject */
public void setSubject(String s)
{
    subject = s;
}
public String[] getSubject()
{
    return subject;
}

/* set/get methods for publications */
public void setPublication(String p)
{
    publication = p;
}
public String getPublication()
{
    return publication;
}
}

/* This class represents all typist staff */
class typist extends staff
{
    int speed;

    /* Input parameters for the constructor are:
       String c: staff code.
       String n: staff name.
       int    s: typist speed.
    */
    public typist(String c, String n, int s)
    {
        super(c, n);
        speed = s;
    }

    /* set/get methods for the typist speed. */
    public void setSpeed(int s)
    {
        speed = s;
    }
    public int getSpeed()
    {
        return speed;
    }
}

/* This class represents all the officer staff */
class officer extends staff
{

```

```

int grade;

/* Constructor input parameters are:
   String c: staff code.
   String n: staff name.
   int    g: officer's grade.
*/
public officer(String c, String n, int g)
{
    super(c, n);
    grade = g;
}

/* set and get methods for the officers' grade */
public void setGrade(int g)
{
    grade = g;
}
public int getGrade()
{
    return grade;
}
}

/* This class represents all the regular typist staff */
class regular extends typist
{
    /* Input parameters for the constructor are:
       String c: staff code.
       String n: staff name.
       int    s: typist speed.
    */
    public regular(String c, String n, int s)
    {
        super(c, n, s);
    }
}

/* This class represents all casual typists */
class casual extends typist
{
    float daily_wages;

    /* Input parameters for the constructor are:
       of a casual typist:
       String c: staff code.
       String n: staff name.
       int    s: typist speed.
       float  w: casual's daily wages.
    */
    public casual(String c, String n, int s, float w)
    {
        super(c, n, s);
        daily_wages = w;
    }

    /* This method set the daily wages for a casual staff */

```

```

    public void setWages(float w)
    {
        daily_wages = w;
    }

    /* This method returns the amount of the daily wages */
    public float getWages()
    {
        return daily_wages;
    }
}

```

2. The annual examination results of 100 students are tabulated as follows:

Roll No.	Subject 1	Subject 2	Subject 3
.	.	.	.
.	.	.	.
.	.	.	.

Write a program to read the data and determine the following:

- Total marks obtained by each student.
- The highest marks in each subject and the Roll No. of the student who secured it.
- The student who obtained the highest total marks.

Sample Answer:

```

class Student
{
    String rollNo;
    float mark[] = new float[3];

    Student(){}

    Student(String num, float m1, float m2, float m3)
    {
        rollNo = num;
        mark[0] = m1;
        mark[1] = m2;
        mark[2] = m3;
    }

    public void setRollNo(String num)
    {
        rollNo = new String(num);
    }
    public void setMark(int i, float m)
    {
        mark[i] = m;
    }
    float total()
    {

```

```

        return (mark[0] + mark[1] + mark[2]);
    }
}

class Report
{
    final static int no_stud = 4;
    final static int no_subj = 3;
    static Student studs[] = new Student[100];

    public static void main (String args[])
    {
        Student highMark[] = new Student[3];
        Student highestTotal=null, stud;

        for (int i=0; i<no_stud; i++){
            stud = new Student();
            System.out.print("Enter student ["+(i+1)+"] Roll
                Numer: ");
            stud.setRollNo(Keyboard.readString());
            System.out.println("");
            for (int j=0; j<no_subj; j++){
                System.out.print("Enter student ["+(i+1)+"]
                    mark for subject["+(j+1)+"]: ");
                stud.setMark(j, Keyboard.readFloat());
                System.out.println("");
            }
            studs[i] = stud;
            if (i == 0){
                for (int j=0; j<no_subj; j++)
                    highMark[j] = stud;
                highestTotal= stud;
            } else {
                for (int k=0; k<no_subj; k++){
                    if (highMark[k].mark[k] <
                        studs[i].mark[k])
                        highMark[k] = studs[i];
                }
                if (highestTotal.total() < studs[i].total())
                    highestTotal = studs[i];
            }
        }
        for (int i=0; i<no_stud; i++)
            System.out.println("Roll No.: "+ studs[i].rollNo +"
                total marks: "+ studs[i].total());
        for (int i=0; i<no_subj; i++)
            System.out.println("The highest mark for
                Subject"+(i+1)+" :\tRoll No.:
                "+highMark[i].rollNo+"\t Mark:
                "+highMark[i].mark[i]);
        System.out.println("The highest Total\t:\t Roll No.:
            "+highestTotal.rollNo+"\tTotla:
            "+highestTotal.total());
    }
}

```