



Introduction to Information Systems

Coursework 1

CS4001NA

Submitted By:

Name: Krishna Adhikari

London Met ID: 19030988

Group: C2

Date: 2020-01-17

Submitted to:

Sushil Paudel

Module Leader

Programming

Table of Contents

1. Introduction.....	1
2. Class Diagram.....	2
3. Pseudocode	3
3.1. Pseudocode of StaffHire Class.....	3
3.2. Pseudocode of PartTimeStaffHire Class.....	4
3.3. Pseudocode of FullTimeStaffHire Class.....	5
4. Method Description	6
4.1. Method Description for StaffHire	6
4.2. Method Description for PartTimeStaffHire	6
4.3. Method Description for FullTimeStaffHire.....	7
5. Testing.....	8
6. Error Detection and Correction.....	14
7. Conclusion	16
Appendix	17
Bibliography	26

Table of Figures

Figure 1 Class Diagram	2
Figure 2 Inspect Full Time Staff Hire	8
Figure 3 Re-inspect Full Time Staff Hire After Staff Hire	9
Figure 4 Inspect Part Time Staff Hire	10
Figure 5 Re inspecting Part Time Staff After Staff Hire.....	10
Figure 6 Inspect Part Time Staff Hire	11
Figure 7 Inspecting Part-time Staff Hire after termination	11
Figure 8 Displaying Details of Full Time Staff Hire	12
Figure 9 Displaying Details of Part Time Staff Hire.....	12
Figure 10 Logical error on Part Time Hire	14
Figure 11 Runtime error code	14
Figure 12 Syntax error on returning value	15
Figure 13 Runtime error output/demonstrate.....	15

1. Introduction

At this current time, Java is one of the most popular programming languages. It was originally developed by Sun microsystems in mid-nineties. Later Oracle acquired it and now maintains the release and support of java programming language. Java is popular because of its one major function i.e. Write Once Run Everywhere which is, we can program java in one system and run the compiled code in other different kind of operating system. Java is widely used in various systems like android, web, utility and other. Google mostly used java to develop android applications. Java officially declare more than three billion devices in this world are running java. (Pankaj, 2019)

This is the coursework that is assigned by module leader as an individual task. Actually, this coursework focuses about one of the most important part of Object-Oriented Programming Language (OOPs) concept, which is Inheritance. We are given to make a staff hire system which contains three class. The first StaffHire class is super class and other two classes PartTimeStaffHire and FullTimeStaffHire are sub class that are inherited from parent class StaffHire.

2. Class Diagram

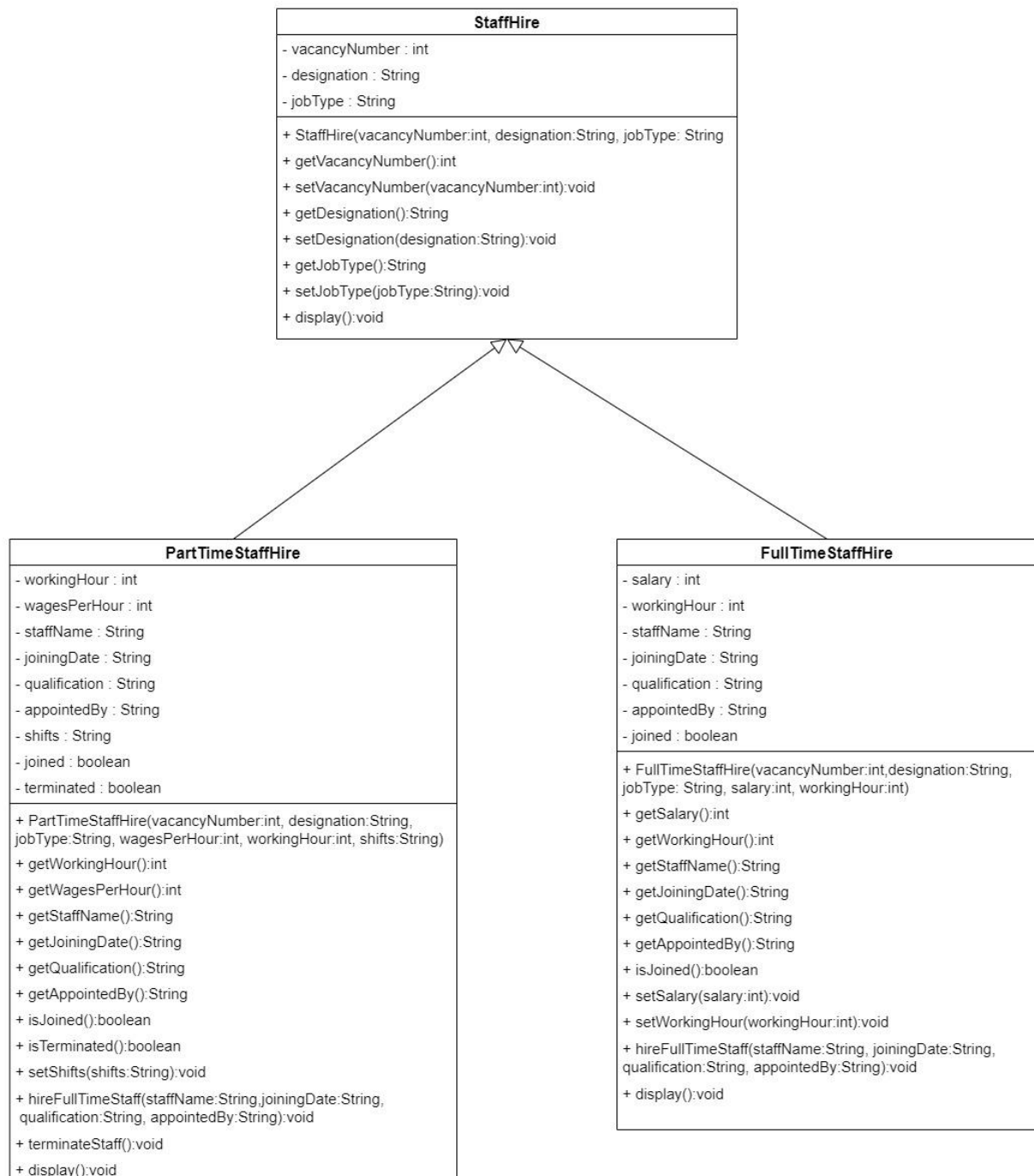


Figure 1 Class Diagram

3. Pseudocode

Pseudocode is a common word in field of programming and algorithm which is a process/technique that allows the developer or programmer to demonstrate the usage of an algorithm and process. Often, algorithm is reflected with the help of pseudo code as they can be interpreted by programmers no matter what their programming background is. According to its name pseudocode is a false code that reflects the program but itself is not a program. It is normally a plain English with common programming constructs. It should be generic. i.e. it shouldn't specify to any language. By looking at these codes a programmer must be able to write code in any languages. (Rouse, 2005)

3.1. Pseudocode of StaffHire Class

Declare Class StaffHire

In class StaffHire declare attributes Vacancy Number, Designation and Job Type with respective data types.

In constructor

StaffHire (Vacancy number, designation, job type) and assign parameter value to attribute.

Use accessor method to return vacancy number

Use mutator method to set new vacancy number

Use accessor method to return designation

Use mutator method to set new designation

Use accessor method to return job type

Use mutator method to set new job type

Use display method to print all the value of attributes suitably

Program End

3.2. Pseudocode of PartTimeStaffHire Class

Declare Class PartTimeStaffHire which is a sub class of StaffHire

In class PartTimeStaffHire declare attributes Salary, Working Hour, Staff Name, Joining Date, Qualification, Appointed By, and Joined.

Accept (Vacancy Number, designation, Job Type, Working Hour, wages per hour, shifts) in parameter then

Pass (Vacancy Number, designation, Job Type) to the constructor of parent class

Assign parameter value to respective attributes then

Staff name = empty string

Joining date = empty string

Qualification = empty string

Appointed by = empty string

Joined = false

Terminated = false

Use accessor method to return workingHour

Use accessor method to return wages per hour

Use accessor method to return staff name

Use accessor method to return joining date

Use accessor method to return qualification

Use accessor method to return person who appoint staff

Use accessor method to return joined status

Use accessor method to return termination status of staff

Declare mutator method for shifts

If the staff is not joined yet then assign attribute value of shifts according to parameter.

Declare method hireparttimestaff(String staff name, String Joining Date, String Qualification, String Appointed By)

If the staff is already appointed print staff name and joining date

else assign attribute value with parameter value

in method terminateStaff

if the staff is already print staff is already appointed

else

staff name = empty string

joining date = empty string

qualification = empty string

appointed by = empty string

joined = false

terminated = true

in display method

call display method of parent class

if the staff is already joined print staff name, wages per hour, working hour, joining date, qualification, appointed by and income per day after calculating.

3.3. Pseudocode of FullTimeStaffHire Class

Declare Class FullTimeStaffHire which is a sub class of StaffHire

Declare attributes salary, working hour, staff name, joining date, qualification, appointed by and joined

In constructor FullTimeStaffHire(vacancy number, designation, job type, salary, working hour)

Call super class constructor by passing (vacancy number, designation, job type)

Assign remaining parameter to attributes

Staff name = empty string

Joining date = empty string

Qualification = empty string

Appointed by = empty string

Joined = false

Define accessor method for salary and return salary

Define accessor method for workingHour and return working hour

Define accessor method for staff name and return staff name

Define accessor method for joining date and return joining date

Define accessor method for qualification and return qualification

Define accessor method for appointed by and return appointed by

Define accessor method for joined and return joined

Define mutator method for salary

If the staff isn't joined yet set salary else

Print developer already appointed and setting salary is not possible.

Define mutator method for workingHour and set working hour to attribute

Declare method hirefulltimestaff(String staff name, String Joining Date, String Qualification, String Appointed By)

If the staff is already appointed, display staff name and joining date

Else

Assign parameter values to attributes and joined = true

In display method

Call display method of super class

If staff is already appointed display staff name, salary, working hour, joining date, qualification and appointed by

Program End

4. Method Description

The collection of statements that is used to perform certain task and return the result are normally said as methods. In java method is called to perform certain tasks. A method specifically performs some sorts of tasks without any returning. In other programming languages, methods are normally said as functions or simply procedures. Normally, methods help us to make code efficient by reducing code redundancy as we can reuse the same code in multiple places without retyping it again and again. (Rai, 2018)

4.1. Method Description for StaffHire

1. `getVacancyNumber()`
This is accessor method that returns vacancyNumber.
2. `setVacancyNumber()`
This is mutator method for vacancy number that is used to set vacancyNumber.
3. `getDesignation()`
This is accessor method for designation that returns designation of staff.
4. `setDesignation()`
This is mutator method for vacancy number that is used to set designation of staff.
5. `getJobType()`
This is accessor method and returns jobType of staff.
6. `setJobType()`
This is mutator method for jobType and is used to set jobType of staff.
7. `display()`
This method displays all value of attributes of StaffHire class.

4.2. Method Description for PartTimeStaffHire

1. `getWorkingHour()`
This is accessor method that returns workingHour.
2. `getWagesPerHour()`
This is accessor method that returns Per hour wage of staff
3. `getStaffName()`
This is accessor method that returns name of staff
4. `getJoiningDate()`
This is accessor method of joiningDate and returns staff's joining date.
5. `getQualification()`
This is accessor method of qualification which returns Qualification of staff.
6. `getAppointedBy()`
This is appointedBy's accessor method that returns name of person who appoint this staff.
7. `isJoined()`
This is accessor method of joined which returns joining status of staff.

8. `isTerminated()`
This is accessor method of terminated which returns termination status of staff.
9. `setShifts()`
This is mutator method for shifts that assign shifts of staff. Whenever joined status is false it will allow to change shifts of staff.
10. `hirePartTimeStaff()`
This method is used to hire part time staff using required parameters. This method will verify joined status of staff. And if the staff is not joined yet it will appoint the staff otherwise it will show detail of the staff.
11. `terminateStaff()`
This method is used to check/change termination status of staff. It will terminate staff only when the staff is not terminated before else it will display suitable message.
12. `display()`
This method is used to display all required information of staff. First, it will call display method of parent class and then print other info of staff only if the staff is joined already.

4.3. Method Description for FullTimeStaffHire

1. `getSalary()`
This is accessor method that returns salary of staff.
2. `getWorkingHour()`
This is accessor method that returns Working Hour of staff.
3. `getStaffName()`
This is accessor method that returns name of staff
4. `getJoiningDate()`
This is accessor method of joiningDate and returns staff's joining date.
5. `getQualification()`
This is accessor method of qualification which returns Qualification of staff.
6. `getAppointedBy()`
This is appointedBy's accessor method that returns name of person who appoint this staff.
7. `isJoined()`
This is accessor method of joined which returns joining status of staff.
8. `setSalary()`
This is mutator method and help to set salary for staff. It will set the salary for staff if the staff is not joined yet otherwise display suitable message.
9. `setWorkingHour()`
This is mutator method for workingHour that assign working hour for staff.
10. `hireFullTimeStaff()`
This method is used to hire full time staff using required parameters. This method will verify joined status of staff. And if the staff is not joined yet it will appoint the staff otherwise it will show detail of the staff.
11. `display()`

This method is used to display all required information of staff. First, it will call display method of parent class and then print other info of staff only if the staff is joined already.

5. Testing

- ❖ Inspect FullTimeStaffHire Class, appoint the full-time staff, and re-inspect the FullTimeStaffHire Class

fullTime1 : FullTimeStaffHire	
private int salary	90000
private int workingHour	7
private String staffName	""
private String joiningDate	""
private String qualification	""
private String appointedBy	""
private boolean joined	false
private int vacancyNumber	101
private String designation	"Product Manager"
private String jobType	"Full Time"

Buttons: Inspect, Get, Show static fields, Close

Figure 2 Inspect Full Time Staff Hire

After creating the object of full-time staff hire class, it is inspected and we are able to see all the values but only some of them are appear which are passed through parameter.

fullTime1 : FullTimeStaffHire

private int salary	90000	Inspect
private int workingHour	7	
private String staffName	"Joseph"	Get
private String joiningDate	"2020-01-01"	
private String qualification	"Java Developer"	
private String appointedBy	"John"	
private boolean joined	true	
private int vacancyNumber	101	
private String designation	"Product Manager"	
private String jobType	"Full Time"	

Show static fields Close

Figure 3 Re-inspect Full Time Staff Hire After Staff Hire

After hiring the full-time staff, we inspected the object again and we are able to view the full-time staff is hired with his/her all details.

- ❖ Inspect PartTimeStaffHire Class, appoint part time staff, and re-inspect the PartTimeStaffHire Class

partTime1 : PartTimeStaffHire

private int workingHour	8	Inspect
private int wagesPerHour	2000	
private String staffName	""	Get
private String joiningDate	""	
private String qualification	""	
private String appointedBy	""	
private String shifts	"Day"	
private boolean joined	false	
private boolean terminated	false	
private int vacancyNumber	901	
private String designation	"SuperVisor"	
private String jobType	"PartTime"	

Show static fields Close

Figure 4 Inspect Part Time Staff Hire

After creating the object of part time staff hire class, we inspect the object of that class and we are able to view few values that are passed using parameters.

partTime1 : PartTimeStaffHire

private int workingHour	8	Inspect
private int wagesPerHour	2000	
private String staffName	"Mark"	Get
private String joiningDate	"2019-08-01"	
private String qualification	"MBA"	
private String appointedBy	"William"	
private String shifts	"Day"	
private boolean joined	true	
private boolean terminated	false	
private int vacancyNumber	901	
private String designation	"SuperVisor"	
private String jobType	"PartTime"	

Show static fields Close

Figure 5 Re inspecting Part Time Staff After Staff Hiring

After hiring the part time staff, we re-inspected the object and we found that part time staff was hired and we can view all the details of that staff.

- ❖ Inspect PartTimeStaffHire Class, change the termination status of a staff, and re-inspect the PartTimeStaffHire Class

partTime1 : PartTimeStaffHire

private int workingHour	8	Inspect Get
private int wagesPerHour	2000	
private String staffName	"Mark"	Close
private String joiningDate	"2019-08-01"	
private String qualification	"MBA"	
private String appointedBy	"William"	
private String shifts	"Day"	
private boolean joined	true	
private boolean terminated	false	
private int vacancyNumber	901	
private String designation	"SuperVisor"	
private String jobType	"PartTime"	
Show static fields		

Figure 6 Inspect Part Time Staff Hire

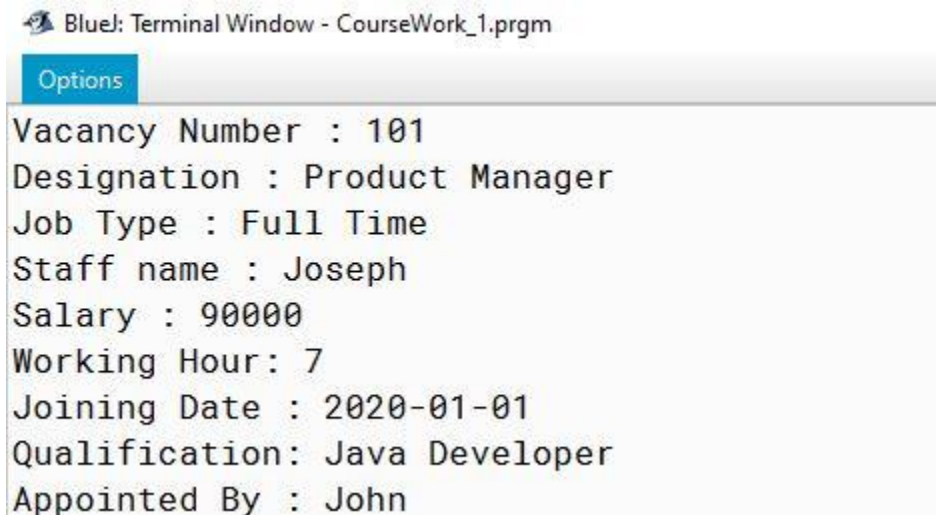
partTime2 : PartTimeStaffHire

private int workingHour	8	Inspect Get
private int wagesPerHour	2000	
private String staffName	""	Close
private String joiningDate	""	
private String qualification	""	
private String appointedBy	""	
private String shifts	"Day"	
private boolean joined	false	
private boolean terminated	true	
private int vacancyNumber	901	
private String designation	"SuperVisor"	
private String jobType	"PartTime"	
Show static fields		

Figure 7 Inspecting Part-time Staff Hire after termination

We then call terminate method and the staff was terminated. After it we re-inspect the object and found that the termination status of the staff is turned to true.

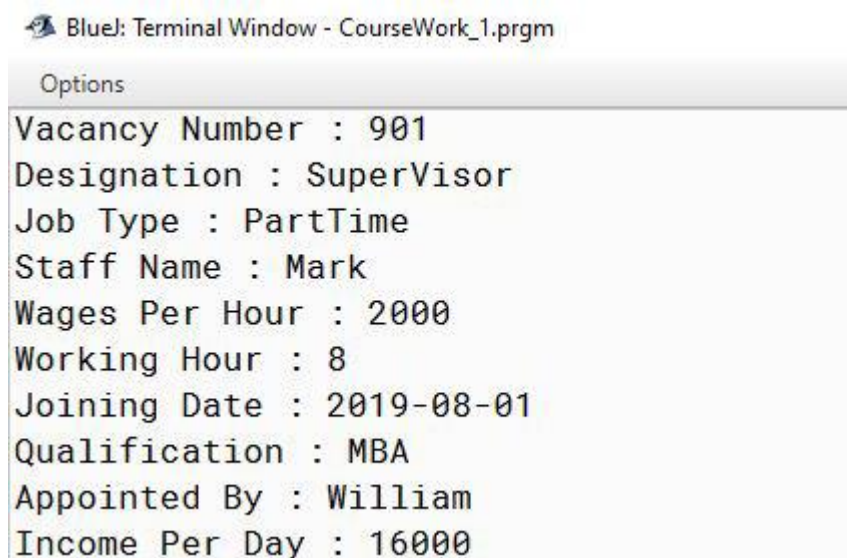
- ❖ Display the detail of FullTimeStaffHire and PartTimeStaffHire Class.



```
BlueJ: Terminal Window - CourseWork_1.prgm
Options
Vacancy Number : 101
Designation : Product Manager
Job Type : Full Time
Staff name : Joseph
Salary : 90000
Working Hour: 7
Joining Date : 2020-01-01
Qualification: Java Developer
Appointed By : John
```

Figure 8 Displaying Details of Full Time Staff Hire

In this test we display all the details of full-time staff hire where we are able to view the vacancy number, name, designation, job type, salary and so on as output.



```
BlueJ: Terminal Window - CourseWork_1.prgm
Options
Vacancy Number : 901
Designation : SuperVisor
Job Type : PartTime
Staff Name : Mark
Wages Per Hour : 2000
Working Hour : 8
Joining Date : 2019-08-01
Qualification : MBA
Appointed By : William
Income Per Day : 16000
```

Figure 9 Displaying Details of Part Time Staff Hire

In this test we display all the details of part-time staff hire where we are able to view the vacancy number, name, designation, job type, salary and so on as output.

6. Error Detection and Correction

❖ Logical Error

```
public void hirePartTimeStaff(String staffName, String joiningDate,
    if (joined == false){
        System.out.println("Name of staff : " + getStaffName());
        System.out.println("Joining Date : " + getJoiningDate());
    } else{
```

Figure 10 Logical error on Part Time Hire

This is the error that didn't affect the running process or compilation process but it doesn't provide the expected result after running the program. During this course work I faced similar problem that I use false in joined status instead of true which result me to get improper result. Although the staff is not joined, the method forces to display staff name and joining date.

❖ Runtime Error on running display method in part time staff hire class

```
//the method below uses same signature as display method in super class and prints
public void display(){
    display();
    if (joined == true){
        System.out.println("Staff Name : " + getStaffName());
        System.out.println("Wages Per Hour : " + getWagesPerHour());
        System.out.println("Working Hour : " + getWorkingHour());
        System.out.println("Joining Date : " + getJoiningDate());
        System.out.println("Qualification : " + getQualification());
        System.out.println("Appointed By : " + getAppointedBy());
        System.out.println("Income Per Day : " + (getWagesPerHour() * getWorkingHour()));
    }
}
```

Figure 11 Runtime error code

```

BlueJ: Terminal Window - CourseWork_1.prgm
Options

Can only enter input while your programming is running

at PartTimeStaffHire.display(PartTimeStaffHire.java:95)
at PartTimeStaffHire.display(PartTimeStaffHire.java:95)
at PartTimeStaffHire.display(PartTimeStaffHire.java:95)
at PartTimeStaffHire.display(PartTimeStaffHire.java:95)
at PartTimeStaffHire.display(PartTimeStaffHire.java:95)
at PartTimeStaffHire.display(PartTimeStaffHire.java:95)
at PartTimeStaffHire.display(PartTimeStaffHire.java:95)
at PartTimeStaffHire.display(PartTimeStaffHire.java:95)
at PartTimeStaffHire.display(PartTimeStaffHire.java:95)
at PartTimeStaffHire.display(PartTimeStaffHire.java:95)
at PartTimeStaffHire.display(PartTimeStaffHire.java:95)
at PartTimeStaffHire.display(PartTimeStaffHire.java:95)
at PartTimeStaffHire.display(PartTimeStaffHire.java:95)
at PartTimeStaffHire.display(PartTimeStaffHire.java:95)
at PartTimeStaffHire.display(PartTimeStaffHire.java:95)
at PartTimeStaffHire.display(PartTimeStaffHire.java:95)
at PartTimeStaffHire.display(PartTimeStaffHire.java:95)
at PartTimeStaffHire.display(PartTimeStaffHire.java:95)
at PartTimeStaffHire.display(PartTimeStaffHire.java:95)
at PartTimeStaffHire.display(PartTimeStaffHire.java:95)

```

Figure 12 Runtime error output/demonstrate

While I tried to display without calling parent display method. When running display method of this class it gives infinite runtime errors. The above image shows the error that I made.

❖ Syntax error while returning value

```

}

public int getStaffName(){ //getter for staffName
    return staffName;
}

public String getJoiningDate(){ //getter for joiningDate
    return joiningDate;
}

```

incompatible types: java.lang.String cannot be converted to int

Figure 13 Syntax error on returning value

This is the error I get while using accessor method in staff name. It is initially of String type but there I tried to return integer value which is not compatible to return. Then I managed it and changed it to String.

7. Conclusion

The language that is used to write computer program is defined as programming language. Java is a high-level programming language developed by Sun Microsystems in nineties of 20th century. It is then used as core component of sun microsystem. Later in 2010 all the java documentation and packages were handled by oracle. Java is used to develop web application, android application and other. In this coursework it helps us to focus the two major concept of object-oriented programming (OOPs) which are inheritance and encapsulation. This makes me clear about those concepts in OOPs.

We are assigned the coursework by module leader. At the beginning, I was in dilemma how to complete this coursework and how to begin. Later, I asked my module leader and he clarify the whole concepts and topics that need to be included in this coursework. At first, I develop pseudocode for the whole three class which then makes me all clear about process of writing program for this coursework. Then I gather my courage and start doing the work. At the coding time, I didn't face much problem during as I already completed basic java course before two years ago. So, I was much more familiar with the concepts and using datatypes, methods and objects in java.

We are assigned to create three class named StaffHire, PartTimeStaffHire and FullTimeStaffHire. Staff Hire class is parent where other two classes were inherited from the parent class and known as sub class or simply child. During creating the classes I thought it is better to create class diagram and I did it. It helped me a lot and makes me clear to step forward to the coding portion. I faced some bugs and errors during coding which I solved discussing with my friends and after critical thinking. Combination of pseudocode and class diagram works better for me to write the code.

Completing this coursework in time is really a hard job as we are given coursework of two different module at same time and they need to be completed and submitted in same time. Although I faced many obstacles and issues during this coursework, I completed this coursework in given period of time with full of time consumption.

Appendix

❖ Appendix for StaffHire Class

```
public class StaffHire{

    private int vacancyNumber;

    private String designation;

    private String jobType;

    //This constructor below initialize the attributes from acquired parameter

    public StaffHire(int vacancyNumber, String designation, String jobType){

        this.vacancyNumber = vacancyNumber;

        this.designation = designation;

        this.jobType = jobType;

    }

    public int getVacancyNumber(){ //getter for vacancy number

        return vacancyNumber;

    }

    public void setVacancyNumber( int vacancyNumber){ //setter for vacancy
number

        this.vacancyNumber = vacancyNumber;

    }

    public String getDesignation(){ //getter for designation

        return designation;

    }

    public void setDesignation( String designation){ //setter for designation

        this.designation = designation;

    }

}
```

```
public String getJobType(){ //getter for job type
    return jobType;
}

public void setJobType(String jobType){ //setter for jobtype
    this.jobType = jobType;
}

public void display(){ //this method provide output of all attributes.
    System.out.println("Vacancy Number : " + getVacancyNumber());
    System.out.println("Designation : " + getDesignation());
    System.out.println("Job Type : " + getJobType());
}
}
```

❖ Appendix for FullTimeStaffHire class

```
public class FullTimeStaffHire extends StaffHire{
    private int salary;
    private int workingHour;
    private String staffName;
    private String joiningDate;
    private String qualification;
    private String appointedBy;
    private boolean joined;

    //This constructor below initialize the attributes from acquired parameter.

    public FullTimeStaffHire(int vacancyNumber, String designation, String
jobType, int salary, int workingHour){
        super(vacancyNumber, designation, jobType);
    }
}
```

```
this.salary = salary;

this.workingHour = workingHour;

this.staffName = "";

this.joiningDate = "";

this.qualification = "";

this.appointedBy = "";

this.joined = false;
}

public int getSalary(){ //getter for salary

    return salary;

}

public int getWorkingHour(){ //getter for workingHour

    return workingHour;

}

public String getStaffName(){ //getter for staffName

    return staffName;

}

public String getJoiningDate(){ //getter of joiningDate

    return joiningDate;

}

public String getQualification(){ //getter for qualification

    return qualification;

}

public String getAppointedBy(){ //getter for appointedBy
```

```
        return appointedBy;
    }

    public boolean isJoined(){ //getter for joined

        return joined;
    }

    public void setSalary( int salary){ //setter for salary

        if(joined == false){

            this.salary = salary;

        } else{

            System.out.println("Developer is already appointed. So, setting salary is
not possible.");

        }

    }

    public void setWorkingHour( int workingHour){ //setter for workingHour

        this.workingHour = workingHour;

    }

    //This method hire full time staff checking their joining status too.

    public void hireFullTimeStaff(String staffName, String joiningDate, String
qualification, String appointedBy){

        if(joined == true){

            System.out.println("Staff Name : " + getStaffName());

            System.out.println("Joining Date : " + getJoiningDate());

        } else {

            this.staffName = staffName;

            this.joiningDate = joiningDate;

        }

    }

}
```

```
        this.qualification = qualification;

        this.appointedBy = appointedBy;

        this.joined = true;
    }
}

//the method below uses same signature as display method in super class
and prints required output

public void display(){
    super.display();

    if(joined == true){
        System.out.println("Staff name : " + getStaffName());
        System.out.println("Salary : " + getSalary());
        System.out.println("Working Hour: " + getWorkingHour());
        System.out.println("Joining Date : " + getJoiningDate());
        System.out.println("Qualification: " + getQualification());
        System.out.println("Appointed By : " + getAppointedBy());
    }
}
}
```

❖ Appendix for PartTimeStaffHire class

```
public class PartTimeStaffHire extends StaffHire{

    private int workingHour;

    private int wagesPerHour;

    private String staffName;

    private String joiningDate;
```



```
private String qualification;

private String appointedBy;

private String shifts;

private boolean joined;

private boolean terminated;


//This constructor below initialize the attributes from acquired parameter

public PartTimeStaffHire(int vacancyNumber, String designation, String
jobType, int workingHour, int wagesPerHour, String shifts){

    super(vacancyNumber, designation, jobType);

    this.workingHour = workingHour;

    this.wagesPerHour = wagesPerHour;

    this.shifts = shifts;

    this.staffName = "";

    this.joiningDate = "";

    this.qualification = "";

    this.appointedBy = "";

    this.joined = false;

    this.terminated = false;

}

public int getWorkingHour(){ //getter for workingHour

    return workingHour;

}

public int getWagesPerHour(){ //getter for wagesPerHour

    return wagesPerHour;
```

```
}

public String getStaffName(){ //getter for staffName
    return staffName;
}

public String getJoiningDate(){ //getter for joiningDate
    return joiningDate;
}

public String getQualification(){ //getter for qualification
    return qualification;
}

public String getAppointedBy(){ // getter for appointedBy
    return appointedBy;
}

public boolean isJoined(){ //getter for joined
    return joined;
}

public boolean isTerminated(){ //getter for termination
    return terminated;
}

public void setShifts(String shifts){ //setter for shifts
    if (joined == false){
        this.shifts = shifts;
    }
}
```

//the below method hire parttime staff by checking their joining status

```
public void hirePartTimeStaff(String staffName, String joiningDate, String
qualification, String appointedBy){

    if (joined == true){

        System.out.println("Name of staff : " + getStaffName());

        System.out.println("Joining Date : " + getJoiningDate());

    } else{

        this.staffName = staffName;

        this.joiningDate = joiningDate;

        this.qualification = qualification;

        this.appointedBy = appointedBy;

        this.joined = true;

        this.terminated = false;

    }

}
```

//the below method is used to terminate the staff

```
public void terminateStaff(){

    if(terminated == true){

        System.out.println("Staff already terminated");

    } else{

        this.staffName = "";

        this.joiningDate = "";

        this.qualification = "";

        this.appointedBy = "";

        this.joined = false;

    }

}
```

```
        this.terminated = true;

    }

}

//the method below uses same signature as display method in super class
and prints required output

public void display(){

    super.display();

    if (joined == true){

        System.out.println("Staff Name : " + getStaffName());

        System.out.println("Wages Per Hour : " + getWagesPerHour());

        System.out.println("Working Hour : " + getWorkingHour());

        System.out.println("Joining Date : " + getJoiningDate());

        System.out.println("Qualification : " + getQualification());

        System.out.println("Appointed By : " + getAppointedBy());

        System.out.println("Income Per Day : " + (getWagesPerHour() *
getWorkingHour()));

    }

}

}
```

Bibliography

Pankaj, 2019. *Java Programming Language*. [Online]
Available at: <https://www.journaldev.com/32975/what-is-java-programming-language>
[Accessed 12 01 2020].

Rai, A., 2018. *Methods in Java*. [Online]
Available at: <https://www.geeksforgeeks.org/methods-in-java/>
[Accessed 12 01 2020].