Coad and Yourdon's six selection characteristics:

Selection Characteristics	Explanation of Characteristics
1. *Retained Information*	the potential object will be useful during analysis only if information about it must be remembered so that the system can function
2. *Needed Services*	the potential object must have a set of identifiable operations that can change the value of its attributes in some way
3. *Multiple Attributes*	during requirements analysis, the focus should be on "major" information (an object with a single attribute may, in fact, be useful during design, but is probably better represented as an attribute of another object during the analysis activity)
4. *Common Attributes*	a set of attributes can be defined for the potential object, and these attributes apply to all occurrences of the object
5. *Common Operations*	a set of operations can be defined for the potential object, and these operations apply to all occurrences of the object
6. *Essential Requirements*	external entities that appear in the problem space and produce or consume information that is essential to the operation of any solution for the system will almost always be defined as objects in the requirements model

Object Oriented Analysis:

POTENTIAL CLASS	RETAINED INFORMATION	NEEDED SERVICES	MULTIPLE ATTRIBUTES	COMMON ATTRIBUTES	COMMON OPERATIONS	ESSENTIAL REQUIREMENTS	ACCEPTED AS CLASS
1. Department	departmentCode, departmentName, regularRate; overtimeRate,	assignment operator , not equal	Department,addDepartmentRecord,viewDepartmentRecord,updateEmployeeRecord	departmentCode	assignment operator , not equal	FileManager, Employee,Main	Department
2. Employee	departmentCode,employeeID,firstName ,lastName ,employeeDepartmentCode ,position,taxRegistrationNumber,nationalInsuranceScheme,dateOfBirth,dateOfHire,hrsWorked	equal, not equal	Employee,addEmployeeRecord,updateEmployeeRecord, viewemployeeRecord,deleteEmployeeRecord	isValidEmployeeCode.	equal, not equal	Department,FileManager, Main	Employee

3. FileManager	-	assignment operator	writeToFile,readFromoFile,updateRecord	readFromFile,writetoFile	assignment operator	Department,Employee,Main	FileManager
4. Payroll	regularPay, overtimePay; grossPay,dateOfProcessing,chequeNumber;	equal , not equal to,greater than, multiplication, assignment operator	setRegularPay,setOvertimePay, calculateRegularPay,calculateOvertimePay,viewDepartmentPayroll	calculateRegularPay.calculateOvertimePay,viewDepartmentPayroll,Department,addDepartmentRecord,viewDepartmentRecord,updateEmployeeRecord	equal to	Employee	Payroll
5. Main		assignment operator,not equal	payrollMenu,employeemenu,departmentMenu, mainMenu	Employee,addEmployeeRecord,updateEmployeeRecord,viewemployeeRecord,deleteEmployeeRecord,	assignment operator	Department,Employee,Payroll	Main
6. GUI	mainMenuPanel,departmentMenuPanel.employeeMenuPanel, payrollMenuPanel	equal to, arrow operator	createMainMenuPanel, createMenuButton,createDepartmentMenuPanel,createEmployeeMenuPanel,createPayrollMenuPanel	createMainMenuPanel,createMenuButton,createDepartmentMenuPanel, createEmployeeMenuPanel,createPayrollMenuPanel	equal to	Main,Department,Employee,Payroll	GUI

The Object Oriented Analysis is an approach used by software developers to understand, define and map requirements a system will need to be fully functional. In this project five Potential Classes were identified, theses classes are:

-
- Department
- Employee
- FileManager
- Payroll
- Main
- GUI

The Department Class is needed to store information about the department, validate employee input and set employee rate, whether it is regular rate or overtime rates. Hence values such as regularRate,overRate,departmentName and departmentCode are needed to fulfill these requirements. The operations needed through this program are the ,not equal to sign (!=) and the assignment operator(=).The assignment operator is used to instantiate occurrences of object classes,arrays as well as boolean values.The data from the Department class will be used in the Employee class since a department is made up of employees(composition relationship),File Manager will also need to store information about each department(department name and department code) , so there will be a direct association between these two classes, and Main will provide the application menu for this class, to add, update and view data in this class.

The Employee Class is required to receive employee information such as, date of birth,first name, last name, NIS, TRN, employee id, position in the department, date of hire as well as hours worked. Operations such as the assignment operator will be needed to format fields that require date, by using Java/C++ defined functions. The not equal to operator will be needed to validate conditions, before the program accept and stores user input , which will lower the debugging time.This employee class would form a composition relation between two other classes, the Payroll class and Department Class, because neither the Payroll Class or Department Class cannot exist without the Employee who makes up the department and has a payroll.

The Payroll Class is needed to calculate and store the regular pay as well as the overtime pay for employees and the date the pay is processed . Numerous operations are required in this class because of the extensive arithmetic needed to calculate employee pay. Multiplication will be used to calculate regular pay by finding the product of the hourly rate and number of hours worked, as well as adding any overtime inclusive of a different overtime rate, to display the total pay. The Payroll class is reliant on the Employee class, because each record from payroll would be needed to be assigned to an employee. The Main class provides the men option to select and modify the Payroll Class, this class would also extend the Employee Class, because of the needed attributes from the parent class such as hours worked, which will be needed in the Payroll class.

FileManager, this class will store all the data that has been validated in the Payroll,Department and Employee Class. This class uses a common attribute with the mentioned classes that will need to serialize the various data types into a singular string , so it can be stored into a file as well as arrays, and evidently it will need to deserialize the stored data to display information to the user. In this class because of the extensive use of files, exception handling has to be done throughout, therefore the not equal sign to validate contents of the file before data is added,deleted or updated is

required. This class will be associated with the Department class, because the Department Class will cover any other relationship with other classes since the employee in particular belongs to the department and the employee has a payroll.
The Main class is fundamental in giving accessibility features to the software, by displaying a menu option to select the needed class of amendment . This will display a menu option to select department ,employee and payroll submenu. Each sub-menu will give direct access to the class of the same of each menu option, that will allow, add, update and a view of stored data. This class will also allow a seamless flow of the algorithm as well as a gracious termination of the program rather than an abrupt finish.

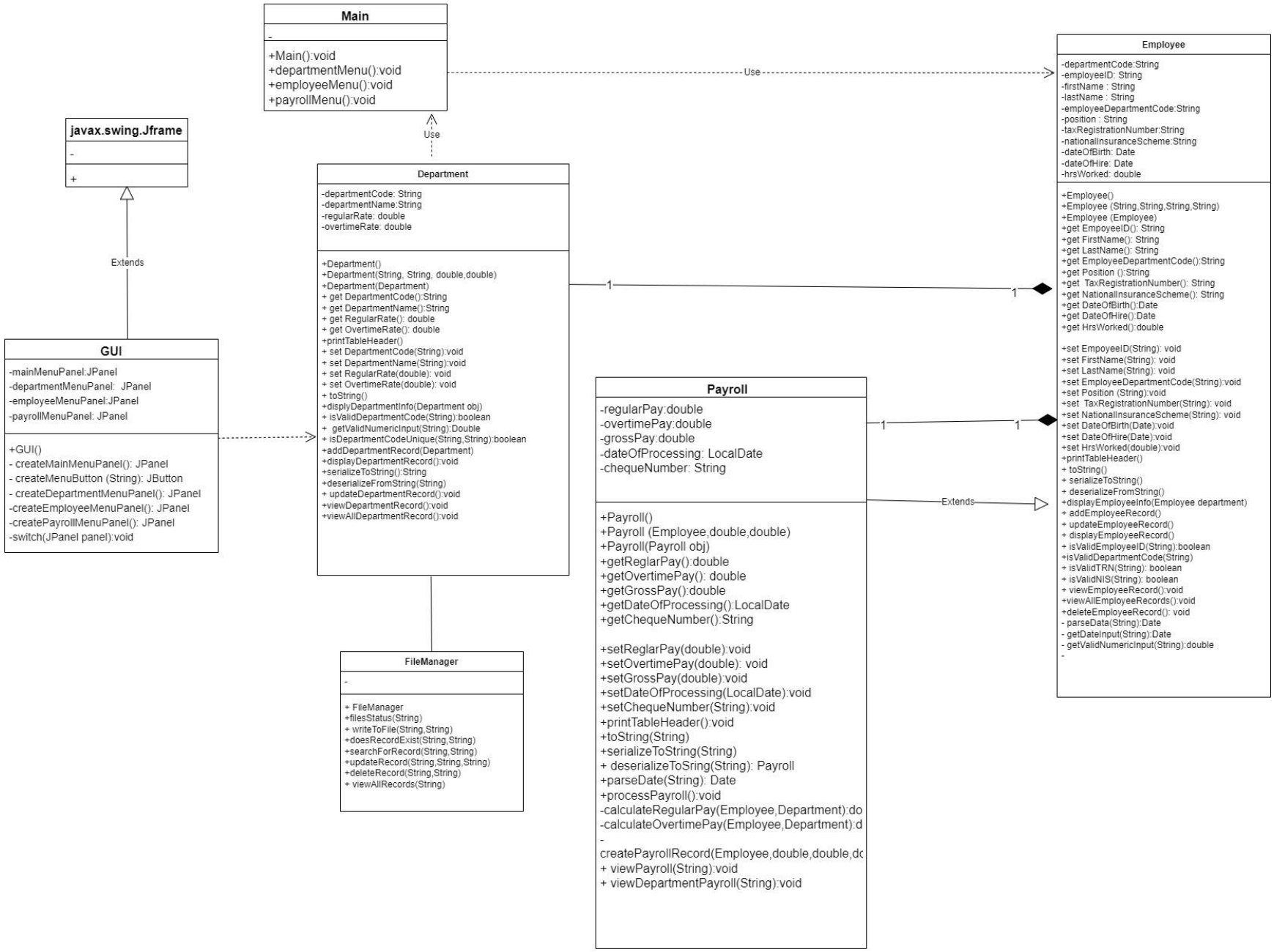
The GUI class is not a functional requirement, because without it the software is still usable, however it is needed for the ease of use by the user as well as the aesthetic display of information. The GUI will be associated with the Main class since, main is the only class that users will directly interact with. If this model is implemented in JAVA, the GUI class will extend the JAVA defined functionality, Jframe that is responsible for the window display, labels, buttons and all other GUI related features.

When this is being implemented in JAVA/C++ , the fundamental principles of OOP will be obeyed these include but are not limited to:

- Variable naming consistency with camel case
- Class naming consistency with pascal case
- Correct Formatting
- Potential Bug Fixes and updates prior to code development and implementation

OBJECT ORIENTED DESIGN

The following outline is displayed in the Object Oriented Design, using UML that displays class attributes, relationships and cardinality.


Main
-
+Main():void
+departmentMenu():void
+employeeMenu():void
+payrollMenu():void
Use
Employee
-departmentCode:String
-employeeID: String
-firstName : String
-lastName : String
-employeeDepartmentCode:String
-position : String
-taxRegistrationNumber:String
-nationalInsuranceScheme:String
-dateOfBirth: Date
-dateOfHire: Date
-hrsWorked: double
+Employee()
+Employee (String,String,String,String)
+Employee (Employee)
+get EmpoyeeID(): String
+get FirstName(): String
+get LastName(): String
+get EmployeeDepartmentCode():String
+get Position ():String
+get TaxRegistrationNumber(): String
+get NationalInsuranceScheme(): String
+get DateOfBirth():Date
+get DateOfHire():Date
+get HrsWorked():double
+set EmpoyeeID(String): void
+set FirstName(String): void
+set LastName(String): void
+set EmployeeDepartmentCode(String):void
+set Position (String):void
+set TaxRegistrationNumber(String): void
+set NationalInsuranceScheme(String): void
+set DateOfBirth(Date):void
+set DateOfHire(Date):void
+set HrsWorked(double):void
+printTableHeader()
+ toString()
+ serializeToString()
+ deserializeFromString()
+displayEmployeeInfo(Employee department)
+ addEmployeeRecord()
+ updateEmployeeRecord()
+ displayEmployeeRecord()
+ isValidEmployeeID(String):boolean
+isValidDepartmentCode(String)
+ isValidTRN(String): boolean
+ isValidNIS(String): boolean
+ viewEmployeeRecord():void
+viewAllEmployeeRecords():void
+deleteEmployeeRecord(): void
- parseData(String):Date
- getDateInput(String):Date
- getValidNumericInput(String):double
-
javax.swing.Jframe
-
+
Extends
Department
-departmentCode: String
-departmentName:String
-regularRate: double
-overtimeRate: double
+Department()
+Department(String, String, double,double)
+Department(Department)
+ get DepartmentCode():String
+ get DepartmentName():String
+ get RegularRate(): double
+ get OvertimeRate(): double
+printTableHeader()
+ set DepartmentCode(String):void
+ set DepartmentName(String):void
+ set RegularRate(double): void
+ set OvertimeRate(double): void
+ toString()
+displyDepartmentInfo(Department obj)
+ isValidDepartmentCode(String):boolean
+ getValidNumericInput(String):Double
+ isDepartmentCodeUnique(String,String):boolean
+addDepartmentRecord(Department)
+displayDepartmentRecord():void
+serializeToString():String
+deserializeFromString(String)
+ updateDepartmentRecord():void
+viewDepartmentRecord():void
+viewAllDepartmentRecord():void
1
1
GUI
-mainMenuPanel:JPanel
-departmentMenuPanel: JPanel
-employeeMenuPanel:JPanel
-payrollMenuPanel: JPanel
+GUI()
- createMainMenuPanel(): JPanel
- createMenuButton (String): JButton
- createDepartmentMenuPanel(): JPanel
-createEmployeeMenuPanel(): JPanel
-createPayrollMenuPanel(): JPanel
-switch(JPanel panel):void
Payroll
-regularPay:double
-overtimePay:double
-grossPay:double
-dateOfProcessing: LocalDate
-chequeNumber: String
+Payroll()
+Payroll (Employee,double,double)
+Payroll(Payroll obj)
+getReglarPay():double
+getOvertimePay(): double
+getGrossPay():double
+getDateOfProcessing():LocalDate
+getChequeNumber():String
+setReglarPay(double):void
+setOvertimePay(double): void
+setGrossPay(double):void
+setDateOfProcessing(LocalDate):void
+setChequeNumber(String):void
+printTableHeader():void
+toString(String)
+serializeToString(String)
+ deserializeToSring(String): Payroll
+parseDate(String): Date
+processPayroll():void
-calculateRegularPay(Employee,Department):do
-calculateOvertimePay(Employee,Department):d
-
createPayrollRecord(Employee,double,double,dc
+ viewPayroll(String):void
+ viewDepartmentPayroll(String):void
1
1
Extends
FileManager
-
+ FileManager
+filesStatus(String)
+ writeToFile(String,String)
+doesRecordExist(String,String)
+searchForRecord(String,String)
+updateRecord(String,String,String)
+deleteRecord(String,String)
+ viewAllRecords(String)