TBSpeed CS101
Design for Project 6

+	Database
+	main(args: String[])

+	Date
+	int: dateMM
+	int: dateDD
+	int: dateYYYY
+	Date(dateMMIn: int, dateDDIn; int, dateYYYYIn:int)
+	getMM(): int
+	getDD(): int
+	getYYYY(): int
+	toString(): String
+	compareTo(otherDate: Date): int

+	Employee

+	Faculty
+	String: name
+	String: address
+	String: phoneNumber
+	String: email
+	String: office
+	String: title
+	String: officeHours
+	Date: hireDate
+	float: salary
+	Faculty(nameIn: String, addressIn: String, phoneIn: String, emailIn: String, officeIn: String, salaryIn: float, hireDateIn: Date, titleIn; String, officeHoursIn:String)
+	toString(): String
+	getDate(): Date
+	getSalary(): float
+	getName(): String

+	GraduateStudent
+	String: name
+	String: address
+	String: phoneNumber
+	String: email
+	String: status
+	String: assistantType
+	GraduateStudent(nameIn: String, addressIn: String, phoneIn: String, emailIn: String, birthDateIn: Date, statusIn:String, assistantTypeIn: String)
+	toString(): String
+	getDate(): Date
+	getAddress(): int
+	getName(): String

+	Person
+	getDate(): Date
+	getAddress(): int
+	getName(): String
+	getSalary(): float

+	Staff
+	String: name
+	String: address
+	String: phoneNumber
+	String: email
+	String: office
+	String: title
+	String: supervisor
+	Date: hireDate

+	float: salary
+	Staff(nameIn: String, addressIn: String, titleIn; String, phoneIn: String, emailIn: String, officeIn: String, salaryIn: float, hireDateIn: Date, supervisorIn:String)
+	toString(): String
+	getDate(): Date
+	getSalary(): float
+	getName(): String

+	Student

+	UndergraduateStudent
+	String: name
+	String: address
+	String: phoneNumber
+	String: email
+	String: status
+	UndergraduateStudent(nameIn: String, addressIn: String, phoneIn: String, emailIn: String, birthDateIn: Date, statusIn:String, assistantTypeIn: String)
+	toString(): String
+	getDate(): Date
+	getName(): String

TBSpeed CS101
Design for Project 6

Data Table for Database

Variable	Type	Use
people	Person[]	Track the people

Data Table for main(String[] args)

Variable	Type	Use
args	String[]	unused
name	String	pass input to the object constructors
address	String	pass input to the object constructors
phoneNumber	String	pass input to the object constructors
email	String	pass input to the object constructors
office	String	pass input to the object constructors
dateStr	String	pass input to the object constructors
title	String	pass input to the object constructors
officeHours	String	pass input to the object constructors
supervisor	String	pass input to the object constructors
status	String	pass input to the object constructors
assistantType	String	pass input to the object constructors
date	Date	pass input to the object constructors
dateMM	int	pass input to the object constructors
dateDD	int	pass input to the object constructors
dateYYYY	int	pass input to the object constructors
salary	float	pass input to the object constructors
personCount	int	Count how many person objects exist
inputFile	File	track the input file
fileScan	Scanner	read the input file
outFile	File	track the output file
fileOutput	PrintStream	print to the output file
lineTotal	int	count the number of lines of input
wastedInput	String	clear the input for next line
temp	Person	hold a temperary person
again	boolean	track weather or not to go again in the while loop
arg1	String	hold arg1 for sorting
arg2	String	hold arg2 for sorting
temp1	Date	hold a temp date for sorting
temp2	Date	hold a temp date for sorting
staffCount	int	Count the number of staff
count	int	count the number of iterations
index	int	count the number of loops
printOut	Person[]	hold the array to be printed
employeeCount	int	hold the number of employees
gradCount	int	hold the number of grad students

Data Table for Date

Variable	Type	Use
dateMM	int	Hold the month of date
dateDD	int	Hold the day of date
dateYYYY	int	Hold the year of date

Data Table for Date(int dateMMIn, int dateDDIn, int dateYYYYIn)

Variable	Type	Use
dateMMIn	int	Hold the month of date comeing in
dateDDIn	int	Hold the day of date comeing in
dateYYYYIn	int	Hold the year of date comeing in

Data Table for getMM()

Variable	Type	Use

Data Table for getDD()

Variable	Type	Use

Data Table for getYYYY()

Variable	Type	Use

Data Table for toString()

Variable	Type	Use
pong	String	Hold the return value

Data Table for compareTo(Date otherDate)

Variable	Type	Use
otherDate	Date	Hold the other date being tested

Data Table for Faculty

Variable	Type	Use
name	String	Hold a faculty object's data
address	String	Hold a faculty object's data
phoneNumber	String	Hold a faculty object's data
email	String	Hold a faculty object's data
office	String	Hold a faculty object's data
title	String	Hold a faculty object's data
officeHours	String	Hold a faculty object's data
hireDate	Date	Hold a faculty object's data
salary	float	Hold a faculty object's data

Data Table for Faculty(String nameIn, String addressIn, String phoneIn, String emailIn, String officeIn, float salaryIn, Date hireDateIn, String titleIn, String officeHoursIn)

Variable	Type	Use
nameIn	String	Initalise a new object
addressIn	String	Initalise a new object
phoneNumberIn	String	Initalise a new object
emailIn	String	Initalise a new object
officeIn	String	Initalise a new object
titleIn	String	Initalise a new object
officeHoursIn	String	Initalise a new object
hireDateIn	Date	Initalise a new object
salaryIn	float	Initalise a new object

Data Table for toString()

Variable	Type	Use
pong	String	Hold the return value

Data Table for getDate()

Variable	Type	Use

Data Table for getSalary()

Variable	Type	Use

Data Table for getName()

Variable	Type	Use

Data Table for GraduateStudent

Variable	Type	Use
name	String	Hold a graduate student object's data
address	String	Hold a graduate student object's data
phoneNumber	String	Hold a graduate student object's data
email	String	Hold a graduate student object's data
status	String	Hold a graduate student object's data
assistantType	String	Hold a graduate student object's data

birthDate	Date	Hold a graduate student object's data

Data Table for GraduateStudent(String nameIn, String addressIn, String phoneIn, String emailIn, Date birthDateIn, String statusIn, String assistantTypeIn)

Variable	Type	Use
nameIn	String	Initalise a new object
addressIn	String	Initalise a new object
phoneNumberIn	String	Initalise a new object
emailIn	String	Initalise a new object
statusIn	String	Initalise a new object
assistantTypeIn	String	Initalise a new object
birthDateIn	Date	Initalise a new object

Data Table for toString()

Variable	Type	Use
pong	String	Hold the return value

Data Table for getDate()

Variable	Type	Use

Data Table for getAddress()

Variable	Type	Use
addressInt	in	hold the int part of the address

Data Table for getName()

Variable	Type	Use

Data Table for getDate() - in Person Class

Variable	Type	Use
pong	Date	Hold the return value

Data Table for getAddress() - in Person Class

Variable	Type	Use
pong	int	Hold the return value

Data Table for getName() - in Person Class

Variable	Type	Use
pong	String	Hold the return value

Data Table for getSalary() - in Person Class

Variable	Type	Use
pong	float	Hold the return value

Data Table for Staff

Variable	Type	Use
name	String	Hold a faculty object's data
address	String	Hold a faculty object's data
phoneNumber	String	Hold a faculty object's data
email	String	Hold a faculty object's data
office	String	Hold a faculty object's data
title	String	Hold a faculty object's data
supervisor	String	Hold a faculty object's data
hireDate	Date	Hold a faculty object's data
salary	float	Hold a faculty object's data

Data Table for Staff(String nameIn, String addressIn, String titleIn, String phoneIn, String emailIn, String officeIn, float salaryIn, Date hireDateIn, String supervisorIn)

Variable	Type	Use
nameIn	String	Initalise a new object
addressIn	String	Initalise a new object

phoneNumberIn	String	Initalise a new object
emailIn	String	Initalise a new object
officeIn	String	Initalise a new object
titleIn	String	Initalise a new object
supervisorIn	String	Initalise a new object
hireDateIn	Date	Initalise a new object
salaryIn	float	Initalise a new object

Data Table for toString()

Variable	Type	Use
pong	String	Hold the return value

Data Table for getDate()

Variable	Type	Use

Data Table for getSalary()

Variable	Type	Use

Data Table for getName()

Variable	Type	Use

Data Table for UndergraduateStudent

Variable	Type	Use
name	String	Hold a graduate student object's data
address	String	Hold a graduate student object's data
phoneNumber	String	Hold a graduate student object's data
email	String	Hold a graduate student object's data
status	String	Hold a graduate student object's data
birthDate	Date	Hold a graduate student object's data

Data Table for UndergraduateStudent(String nameIn, String addressIn, String phoneIn, String emailIn, Date birthDateIn, String statusIn)

Variable	Type	Use
nameIn	String	Initalise a new object
addressIn	String	Initalise a new object
phoneNumberIn	String	Initalise a new object
emailIn	String	Initalise a new object
statusIn	String	Initalise a new object
birthDateIn	Date	Initalise a new object

Data Table for toString()

Variable	Type	Use
pong	String	Hold the return value

Data Table for getDate()

Variable	Type	Use

Data Table for getName()

Variable	Type	Use

TBSpeed CS101
Design for Project 6

```
Algorithm for main(String[] args)
String name, address, phoneNumber, email, office, dateStr, title, officeHours, supervisor, status, assistantType
Date date
int dateMM, dateDD, dateYYYY
float salary
people <- Person[0..99]
personCount <- 0
File inputFile <- new File(args[0])
Scanner fileScan <- new Scanner(inputFile)
File outFile <- new File(args[1])
PrintStream fileOutput <- new PrintStream(outFile)
fileOutput.println("Project 6\r\nThomas Belloli - CS 101-02\r\nThe next lines contain an echo of the input file")
lineTotal <- 0
while (fileScan.hasNextLine())
    fileOutput.println("\t" + fileScan.nextLine())
    lineTotal++
inputFile <- new File(args[0])
fileScan <- new Scanner(inputFile)
fileScan.useDelimiter("#|\\n")
String wastedInput
while (personCount < lineTotal)
    switch (fileScan.next()) :
        case "u":
            name <- fileScan.next()
            address <- fileScan.next()
            phoneNumber <- fileScan.next()
            email <- fileScan.next()
            dateStr <- fileScan.next()
            dateMM <- Integer.parseInt(dateStr.substring(0, 2))
            dateDD <- Integer.parseInt(dateStr.substring(3, 5))
            dateYYYY <- Integer.parseInt(dateStr.substring(6, 10))
            date <- new Date(dateMM, dateDD, dateYYYY)
            status <- fileScan.next()
            people[personCount] <- new UndergraduateStudent(name, address, phoneNumber, email, date, status)
            personCount++
            if (personCount <= 19)
                wastedInput <- fileScan.next()
            break
        case "g":
            name <- fileScan.next()
            address <- fileScan.next()
            phoneNumber <- fileScan.next()
            email <- fileScan.next()
            dateStr <- fileScan.next()
            dateMM <- Integer.parseInt(dateStr.substring(0, 2))
            dateDD <- Integer.parseInt(dateStr.substring(3, 5))
            dateYYYY <- Integer.parseInt(dateStr.substring(6, 10))
            date <- new Date(dateMM, dateDD, dateYYYY)
            status <- fileScan.next()
            assistantType <- fileScan.next()
            people[personCount] <- new GraduateStudent(name, address, phoneNumber, email, date, status, assistantType)
```

```
                personCount++
                if (personCount <= 19)
                    wastedInput <- fileScan.next()
                break
            case "f":
                name <- fileScan.next()
                address <- fileScan.next()
                phoneNumber <- fileScan.next()
                email <- fileScan.next()
                office <- fileScan.next()
                salary <- fileScan.nextFloat()
                dateStr <- fileScan.next()
                dateMM <- Integer.parseInt(dateStr.substring(0, 2))
                dateDD <- Integer.parseInt(dateStr.substring(3, 5))
                dateYYYY <- Integer.parseInt(dateStr.substring(6, 10))
                date <- new Date(dateMM, dateDD, dateYYYY)
                title <- fileScan.next()
                officeHours <- fileScan.next()
                people[personCount] <- new Faculty(name, address, phoneNumber, email, office, salary, date, title, officeHours)
                personCount++
                if (personCount <= 19)
                    wastedInput <- fileScan.next()
                break
            case "s":
                name <- fileScan.next()
                address <- fileScan.next()
                title <- fileScan.next()
                phoneNumber <- fileScan.next()
                email <- fileScan.next()
                office <- fileScan.next()
                salary <- fileScan.nextFloat()
                dateStr <- fileScan.next()
                dateMM <- Integer.parseInt(dateStr.substring(0, 2))
                dateDD <- Integer.parseInt(dateStr.substring(3, 5))
                dateYYYY <- Integer.parseInt(dateStr.substring(6, 10))
                date <- new Date(dateMM, dateDD, dateYYYY)
                supervisor <- fileScan.next()
                people[personCount] <- new Staff(name, address, title, phoneNumber, email, office, salary, date, supervisor)
                personCount++
                if (personCount <= 19)
                    wastedInput <- fileScan.next()
                break
Person temp
again <- true
String arg1, arg2
Date temp1, temp2
while (again)
    for index <- 0 loop till index < personCount - 1 by index++ each step
        arg1 <- people[index].getName()
        arg2 <- people[index + 1].getName()
        if (-1 == arg1.compareTo(arg2))
            temp <- people[index]
            people[index] <- people[index + 1]
            people[index + 1] <- temp
            again <- false
    if (NOT again)
        again <- true
    else
        again <- false
fileOutput.println("\r\nDatabase Printout, sorted by name")
for index <- 0 loop till index < personCount by index++ each step
    fileOutput.println(people[index])
staffCount <- 0
count <- 0
again <- true
for index <- 0 loop till index <= personCount by index++ each step
    if (people[index] instanceof Staff)
        staffCount++
Person[] printOut <- Person[0..staffCount-1]
for index <- 0 loop till index <= personCount by index++ each step
    if (people[index] instanceof Staff)
        printOut[count] <- people[index]
        count++
while (again)
    for index <- 0 loop till index < staffCount - 1 by index++ each step
        temp1 <- printOut[index].getDate()
        temp2 <- printOut[index + 1].getDate()
        if (-1 == temp1.compareTo(temp2))
            temp <- printOut[index]
            printOut[index] <- printOut[index + 1]
            printOut[index + 1] <- temp
            again <- false
    if (NOT again)
        again <- true
    else
        again <- false
fileOutput.println("\r\nStaff Printout, sorted by hire date")
for index <- 0 loop till index < printOut.length by index++ each step
    fileOutput.println(printOut[index])
count <- 0
again <- true
employeeCount <- 0
for index <- 0 loop till index <= personCount by index++ each step
    if (people[index] instanceof Employee)
        employeeCount++
printOut <- Person[0..employeeCount-1]
for index <- 0 loop till index <= personCount by index++ each step
    if (people[index] instanceof Employee)
        printOut[count] <- people[index]
        count++
while (again)
    for index <- 0 loop till index < employeeCount - 1 by index++ each step
    if (printOut[index].getSalary() < printOut[index + 1].getSalary())
        temp <- printOut[index]
        printOut[index] <- printOut[index + 1]
        printOut[index + 1] <- temp
        again <- false
    if (NOT again)
```

```
            again <- true
        else
            again <- false
fileOutput.println("\r\nEmployee Printout, sorted by salary")
for index <- 0 loop till index < printOut.length by index++ each step
    fileOutput.println(printOut[index])
count <- 0
again <- true
gradCount <- 0
for index <- 0 loop till index <= personCount by index++ each step
    if (people[index] instanceof GraduateStudent)
        gradCount++
printOut <- Person[0..gradCount-1]
for index <- 0 loop till index <= personCount by index++ each step
    if (people[index] instanceof GraduateStudent)
        printOut[count] <- people[index]
        count++
while (again)
    for index <- 0 loop till index < gradCount - 1 by index++ each step
        if (printOut[index].getAddress() < printOut[index + 1].getAddress())
            temp <- printOut[index]
            printOut[index] <- printOut[index + 1]
            printOut[index + 1] <- temp
            again <- false
    if (NOT again)
        again <- true
    else
        again <- false
    fileOutput.println("\r\nGraduate Student Printout, sorted by address")
    for index <- 0 loop till index < printOut.length by index++ each step
        fileOutput.println(printOut[index])
```

```
Algorithm for toString()
String pong <- ""
switch (dateMM) :
    case 1:
        pong <- "January"
        break
    case 2:
        pong <- "February"
        break
    case 3:
        pong <- "March"
        break
    case 4:
        pong <- "April"
        break
    case 5:
        pong <- "May"
        break
    case 6:
        pong <- "June"
        break
    case 7:
        pong <- "July"
        break
    case 8:
        pong <- "August"
        break
    case 9:
        pong <- "September"
        break
    case 10:
        pong <- "October"
        break
    case 11:
        pong <- "November"
        break
    case 12:
        pong <- "December"
        break
pong <- pong + " " + dateDD + ", " + dateYYYY
return pong
```

Algorithm for compareTo(Date otherDate)

```
if (dateYYYY > otherDate.getYYYY())
    return 1
else if (dateYYYY < otherDate.getYYYY())
    return -1
else
    if (dateMM > otherDate.getMM())
        return 1
    else if (dateMM < otherDate.getMM())
        return -1
    else
        if (dateDD > otherDate.getDD())
            return 1
        else if (dateDD < otherDate.getDD())
            return -1
        else
            return -5
```

Algorithm for toString()

```
    String pong <- "Faculty"
    pong <- pong + "\r\n\tname: " + name
    pong <- pong + "\r\n\taddress: " + address
    pong <- pong + "\r\n\tphone number: " + phoneNumber
    pong <- pong + "\r\n\te-mail address: " + email
    pong <- pong + "\r\n\toffice: " + office
    pong <- pong + "\r\n\tsalary: " + salary
    pong <- pong + "\r\n\thire date: " + hireDate
    pong <- pong + "\r\n\ttitle: " + title
    pong <- pong + "\r\n\toffice hours: " + officeHours
    return pong
```

Algorithm for toString()

```
    String pong <- "Graduate Student"
    pong <- pong + "\r\n\tname: " + name
    pong <- pong + "\r\n\taddress: " + address
    pong <- pong + "\r\n\tphone number: " + phoneNumber
    pong <- pong + "\r\n\te-mail address: " + email
    pong <- pong + "\r\n\tbirth date: " + birthDate
    pong <- pong + "\r\n\tstatus: " + status
    pong <- pong + "\r\n\tassistantship type: " + assistantType
    return pong
```

Algorithm for toString()

```
    String pong <- "Staff"
    pong <- pong + "\r\n\tname: " + name
    pong <- pong + "\r\n\taddress: " + address
    pong <- pong + "\r\n\ttitle: " + title
    pong <- pong + "\r\n\tphone number: " + phoneNumber
    pong <- pong + "\r\n\te-mail address: " + email
    pong <- pong + "\r\n\toffice: " + office
    pong <- pong + "\r\n\tsalary: " + salary
    pong <- pong + "\r\n\thire date: " + hireDate
    pong <- pong + "\r\n\tsupervisor: " + supervisor
    return pong
```

Algorithm for toString()

```
    String pong <- "Undergraduate Student"
    pong <- pong + "\r\n\tname: " + name
    pong <- pong + "\r\n\taddress: " + address
    pong <- pong + "\r\n\tphone number: " + phoneNumber
    pong <- pong + "\r\n\te-mail address: " + email
    pong <- pong + "\r\n\tbirth date: " + birthDate
    pong <- pong + "\r\n\tstatus: " + status
    return pong
```