COSC1337 Programming Fundamentals II /ITSE2357 Advanced Object-Oriented Programming

The submission opens two days before the due day, and closes right at the due time on due day. One submission is allowed per project. It is students' responsibilities to submit timely and correctly in order to get credits. Students MUST start each project when it is first discussed in class in order to complete it on time. All projects are individual projects. Project due days may be changed to better align with lectures.

If one of the following is true, not credits will be given:

- The project is late.
- The algorithm or class design is missing.
- The project has syntax errors.
- There are no comments (Javadoc format required) at all in your code.
- Wrong files are submitted.
- **A project is a copy and modification of another student's project. (Both will receive 0.)**

Files to be submitted through Canvas:

- The UML design (UML class diagram)
- All source codes and driver programs
- All supporting files if any

Software Development Project Rubric: (Design is worth 20%; the rest is worth 80 %.)
Analysis: There will be no credit if the software doesn't meet customer specification at all.

- Does the software meet the exact customer specification?
- Does the software read the exact input data and display the exact output data as they are shown in sample runs?
- Does each class include all corresponding data and functionalities?

Design: There will be no credit if the design is missing.

- Is the design (a UML class diagram) an efficient solution?
- Is the design created correctly?

Code: There will be no credit if there are syntactic errors.

- Are there errors in the software?
- Are code conventions and name conventions followed?
- Does the software use the minimum computer resource (computer memory and processing time)?
- Is the software reusable?

Debug:

- Are there bugs in the software?

Documentation: There will be no credit if comments are not included.

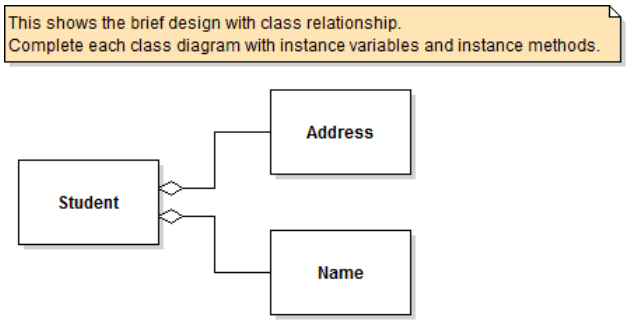
- Are there enough comments included in the software?
- Class comments must be included before a class header.
- Method comments must be included before a method header.
- More comments must be included inside each method.
- All comments must be written in Javadoc format.

Project 1 Writing Classes **Due: See calendar**

Write a class *Student.java* that describes a student with a name, a home address, and a school address. Complete the class with methods that describe a student's behaviors.

Instance variables:

- `name`: The name of this student. The data type is type *Name*.
- `homeAddress`: The home address of this student. The data type is type *Address*.
- `schoolAddress`: The school address of this student. The data type is type *Address*.



Instance methods:

- `public Student()`
  Constructs a student with a default name, a default home address, and a default school address.
- `public Student(Name name, Address homeAddress, Address schoolAddress)`
  Constructs a student with a name, a home address, and a school address.
- `public Name getName()`
  Returns the name of this student.
- `public void setName(Name name)`
  Changes the name of this student.
- `public Address getHomeAddress()`
  Returns the home address of this student.
- `public void setHomeAddress(Address homeAddress)`
  Changes the home address of this student.
- `public Address getSchoolAddress()`
  Returns the school address of this student.
- `public void setSchoolAddress(Address homeAddress)`
  Changes the school address of this student.
- `public String toString()`
  Returns a string representation of this student. The string should be a combination of following three parts:
  - String representation of the name of this student
  - String representation of the home address of this student
  - String representation of the school address of this student
- `public boolean equals(Object obj)`
  Indicates if this student is "Equal to" some other object. If some other object is not a student, return false to indicate not equal to relationship. If some other object is a student, this student is equal to the other student if only if their names, their home addresses, and their school addresses are equal.

Write a program *TestStudent.java* to test your program. *Address.java* and *Name.java* are provided on next page. Place into another package called *name*; place into another package called *address*. Import them into *student* package.

```
/**
 * Represents a street address.
 * @author Qi Wang
 * @version 1.1
 */
public class Address{
        /**
         * The street of this address
         */
        private String street;
        /**
         * The city of this address
         */
        private String city;
        /**
         * The state of this address
         */
        private String state;
        /**
         * The zip code of this address
         */
        private long zipCode;

        /**
         * Constructs a new address with a street name, a city, a state, and a zip code.
         * @param street The street of this address
         * @param city The city of this address
         * @param state The state of this address
         * @param zipCode The zip code of this address
         */
        public Address (String street, String city, String state, long zipCode){
           this.street = street;
           this.city = city;
           this.state = state;
           this.zipCode = zipCode;
        }

        /**
         * Changes the street of this address.
         * @param street A string literal specifying the street of this address
         */
        public void setStreet(String street){
               this.street = street;
        }

        /**
         * Returns the street of this address.
         * @return A string literal specifying the street of this address
         */
        public String getStreet(){
               return this.street;
        }



        /**
         * Returns the city of this address.
         * @return A string literal specifying the city of this address
         */
       public String getCity() {
              return city;
       }

       /**
        * Changes the city of this address.
        * @param city  A string literal specifying the city of this address
        */
```

```
	public void setCity(String city) {
		this.city = city;
	}

	/**
	 * Returns the state of this address
	 * @return A string literal specifying the state of this address
	 */
	public String getState() {
		return state;
	}

	/**
	 * Changes the state of this address.
	 * @param state  A string literal specifying the state of this address
	 */
	public void setState(String state) {
		this.state = state;
	}

	/**
	 * Returns the zip code of this address.
	 * @return A long number specifying the zip code of this address
	 */
	public long getZipCode() {
		return zipCode;
	}

	/**
	 * Changes the zip code of this address.
	 * @param zipCode  A number specifying the zip code of this address
	 */
	public void setZipCode(long zipCode) {
		this.zipCode = zipCode;
	}

	/**
	  * Returns a string representation of this Address.
	  * @return A string literal specifying this address
	  */
	 public String toString(){
	    String result;

	    result = this.street + "\n";
	    result += this.city + ", " + this.state + "  " + this.zipCode;

	    return result;
	 }

	 /**
	  * Indicates if this address is "equal to" some other object. If some other object is not an
	  * address, return false to indicate not equal to relationship. If some other object is an
	  * address, they are equal objects if their streets, cities, states, and zip codes are
	  * equal.
	  * @param obj A reference to some other object
	  * @return A boolean specifying if this address is "equal to" some other object        */
	 public boolean equals(Object obj){
		if(!(obj instanceof Address)){
			return false;

		}

		Address other = (Address)obj;
		return this.street.equalsIgnoreCase(other.street) &&
			this.city.equalsIgnoreCase(other.city) &&
			this.state.equalsIgnoreCase(other.state) && this.zipCode == other.zipCode;
	 }
}
```

```java
/**
 * Represents a name with first name, middle name and last name.
 *
 * @author Qi Wang
 * @version 1.1
 *
 */
public class Name {
	/**
	 *  The first name of the name
	 */
	private String first;
	/**
	 * The middle name of the name
	 */
	private String middle;
	/**
	 * The last name of the name
	 */
	private String last;

	/**
	 * Constructs a name with a first name, a middle name, an a last name.
	 *
	 * @param first The first name of this name
	 * @param middle The middle name of this name
	 * @param last The last name of of this name
	 */
	public Name(String first , String middle , String last ){
		this.first = first.trim();
		this.middle = middle.trim();
		this.last = last.trim();
	}

	/**
	 * Constructs a name with a default first name, a default middle name, and a default
	 * last name.
	 */
	public Name(){
		this.first = "";
		this.middle = "";
		this.last = "";
	}

	/**
	 * Returns the first name of this name.
	 *
	 * @return A String literal specifying the first name of this name
	 */
	public String getFirst(){
		return this.first;
	}

	/**
	 * Changes the first name of this name.
	 *
	 * @param first The first name of this name
	 */
	public void setFirst(String first){
		this.first = first;
	}

	/**
	 * Returns the middle name of the name.
	 * @return A String literal specifying the middle name of the name
	 */
	public String getMiddle(){
		return this.middle;
```

```
}

/**
 * Changes the middle name of this name.
 *
 * @param middle The middle name of this name
 */
public void setMiddle(String middle){
       this.middle = middle;
}

/**
 * Returns the last name of this name.
 *
 * @return A String literal specifying the last name of the name
 */
public String getLast(){
       return this.last;
}

/**
 * Changes the last name of this name.
 *
 * @param last The last name of this name
 */
public void setLast(String last){
       this.last = last;
}

/**
 * Returns the initials in upper case.
 *
 * @return A string literal specifying the initials in upper case
 */
public String initials(){
       if (this.middle.isEmpty()){
              return (this.first.substring(0,1) + this.last.substring(0,1)).toUpperCase();
       }else{
          return (this.first.substring(0,1) + this.middle.substring(0,1) +
                  this.last.substring(0,1)).toUpperCase();
       }
}
/**
 * Returns the length of the full name.
 *
 * @return A integer specifying the length of this name
 */
public int length(){
       return this.first.length() + this.middle.length() + this.last.length();
}

/**
 * Returns a String representation of this name.
 *
 * @return A String with a name's last name, first name, and middle name initial
 */
public String toString(){
       return getClass().getSimpleName() + ": " +
            this.last + ","  +  this.first  +   " " +  this.middle.substring(0,1) + ".";
}

/**
 * Indicates if this name is "equal to" some other object. If some other object is not a
 * name, return false to indicate "not equal to" relationship. If some other object
 * is a name, they are equal if their last name, first name, and middle names are equal.
 *
 * @param Obj A reference to some other object
 * @return A boolean value specifying if this name is "equal to" some other object
```

```
         */
        public boolean equals(Object Obj){
                if (!(Obj instanceof Name)){
                        return false;
                }
                //compare this name with other name
                Name other = (Name)Obj;

                //compare three parts individually
                return this.first.equalsIgnoreCase(other.first) &&
                        this.middle.equalsIgnoreCase(other.middle) &&
                        this.last.equalsIgnoreCase(other.last);
        }


}
```