# Unit 5 Assignment

## GENERAL

1. Save each class in a separate Java source code file and name it *StuPrefixClassName.java*. Follow the coding style as explained in Unit 1 Assignment.

2. Overall comment your program appropriately (file prolog comment, method prolog comment, and comments on the code). Pay attention to the standard stuff like coding style, meaningful identifier names, indention, and locations of braces.

3. When you’re done with an exercise, include a screenshot of the execution of the program in your assignment report document. This is to remind you to verify the execution result.

## EXERCISES

We will practice inheritance and composition in this assignment. All classes from this assignment should be created under the same Project in your IDE.

1. Use Package for Your Project

Java uses packages to organize related classes. Packages are just like folders you use on your computer. The import statements we’ve been using such as import java.util.ArrayList means ArrayList class is listed under a package called util, which in turn is listed under package java.

Here we will practice how to create and use a package for our own project. The instructions below are for Eclipse and VS Code on a Windows machine. Please consult the documentation or Google if you use a different IDE or operating system (like Mac).

|  |  |
| --- | --- |
| **Instructions for Eclipse on a Windows machine** | **Instructions for VS Code on a Windows machine** |
| Create your project as before and accept default options at “Create Java Project” window:  Title: Project Layout Option in Create Java Project window - Description: Use the default "Create separate folders for sources and class files" option. | Create a Java project using command “Java: Create Java Project…”. You can search for the command from Help > Show all commands.    Choose “No build tools” for the project type.    Choose a location i.e. folder for the project.    Next name your project, such as Unit5:    Delete the default App.java file by right clicking on the file and choose “Delete” from the pop-up menu: |
| Next create a package before adding any classes into your project. | |
| Select your project and then File menu 🡪 New 🡪 package:    Name the package with your last name. For example, student John Doe will name his package doe. | Add a new folder under “src” (which will be your package) by right clicking on the “src” folder and choosing “New Folder” command.    Name the folder (i.e. package) with your last name. For example, student John Doe will name his package doe. |
| Your project will look like this in Package Explorer in Eclipse: | After you’re done typing, it will look like this: |
| Now each time you add a new Java class to this project it will be automatically created under this package unless you specify otherwise. A line of java code package *your\_package\_name*; will be added automatically as the first statement in each .java file of this project.  A package defined in a Java project corresponds to a file folder on your computer. If you open your project in Windows Explorer, you will see a folder named after your package under src folder. All .java files are now inside of this package folder. Please include a similar screenshot in your assignment report at the end of the project as a proof of completing the package step.    (Content of the package folder in Windows Explorer) | |

1. (Name this class StuPreEmployee) Find an Employee class in an Assignment5Code.txt for this assignment. Put the code in a class called StuPreEmployee. Make sure there is no syntax error in this class before moving to the next exercise.

Answer this question: why is this class abstract (i.e. would it work to not declare the class as abstract)?

1. (Name this class StuPreStaff) Add a Staff class as a subclass of the Employee class based on the following specification:

|  |
| --- |
| **StuPreStaff** |
|  |
| + StuPreStaff()  + StuPreStaff(id : String, firstName : String, lastName : String, position : String, hiringDate : LocalDate, annualSalary : int)  + getAnnualBonus() : int |

* Default constructor: constructs an object and initialize fields to the default values of their types.
* The parameterized constructor: constructs an object with the given values.
* getAnnualBonus(): returns the annual bonus of a staff. The annual bonus for a staff member is 7.5% of the annual salary.

You will use a new class in this exercise: LocalDate

import java.time.LocalDate;

<https://docs.oracle.com/en/java/javase/11/docs/api/java.base/java/time/LocalDate.html>

1. (Name this class StuPreTestHr) Add a TestHr class with an empty main(). It will serve as the client program.

Add the following to your main() :

1). Create a Staff object with your name and some fake information. Add “S1” to the first name as it will help verifying results later when we add a staff to a manager’s team. Follow this example:

StuPreStaff s1 = **new** StuPreStaff("101", "S1Bob", "Ellen", "Technician I", LocalDate.*now*().minusYears(2), 80000);

2). Create a 2nd Staff object with the information of a friend of yours. Add “S2” to its first name. Use a different hiring date like one year before the now date.

3). Print out both objects. The print of the sample object given above may look like this:

ID: 101, Name: S1Bob Ellen, Position: Technician I, HiringDate: 2019-09-22, Salary: $80,000 (Bonus: $6,000)

1. (Name this class StuPreManager) Add a Manager class as a subclass of your Staff class (!!! Staff class, not Employee class) based on the following specification:

|  |
| --- |
| **StuPreManager** |
|  |
| + StuPreManager()  + StuPreManager(id : String, firstName : String, lastName : String, position : String, hiringDate : LocalDate, annualSalary : int)  + getAnnualBonus() : int |

* Default constructor: construct an object and initialize fields to the default values of their types.
* The parameterized constructor: constructs an object with the given values.
* getAnnualBonus(): returns the annual bonus of a manager. The annual bonus for a manager is 10% of the annual salary.

Add the following to the main() of your TestHr:

Create a staff obj #1

Create a staff obj #2

(add) Create a Manager obj #1: with your instructor’s name and some fake information. Add “M1” to first name. Follow this example:

StuPreManager m1 = **new** StuPreManager("201", "M1John", "Doe", "Tech Lead II", LocalDate.*now*().minusYears(4), 150000);

(add) Create a Manager obj #2 with some different information. Add “M2” to first name.

Print staff #1

Print staff #2

(add) Print manager #1

(add) Print manager #2

The print from a Manager object should look similar to that of a staff object at this point as they currently invoke the same toString().

1. (class StuPreStaff) Add those additional members (highlighted) and update some existing members:

|  |
| --- |
| **StuPreStaff** |
| - manager : StuPreManager |
| + StuPreStaff()  + StuPreStaff(id : String, firstName : String, lastName : String, position : String, hiringDate : LocalDate, annualSalary : int)  + getManager() : StuPreManager  # setManager(manager : StuPreManager) : void  + toString() : String  + getAnnualBonus() : int |

* The additional data member identifies the manager of this staff.
* The parameterized constructor: add a statement to initialize the manager field to null.
* getXXX() and setXXX() methods follow the general convention of such methods.
  + setManager() is a protected method (# means protected).
* toString(): returns a string representation of the calling object in this format:

*str-returned-by-toString() of its super*, Manager: full-*name-of-manager*

// if this staff has a manager

Or

*str-returned-by-toString() of its super*, Manager: none

// this staff doesn’t have any manager (i.e. its manager field is null)

Rerun your program. The print of each object should show an additional portion “, Manager: none”

1. (class StuPreManager) Add those additional members (highlighted) and update some existing members:

|  |
| --- |
| **StuPreManager** |
| - teamMembers : ArrayList<StuPreEmployee> |
| + StuPreManager()  + StuPreManager(id : String, firstName : String, lastName : String, position : String, hiringDate : LocalDate, annualSalary : int)  + addTeamMember(member : StuPreStaff) : boolean  + removeTeamMember(member : StuPreStaff) : boolean  + toString() : String  + getAnnualBonus() : int |

* The additional data member tracks the employees managed by this manager.
* Both constructors: add a statement to each constructor to initialize the ArrayList data member to an empty ArrayList object (!!! not null).
* addTeamMember(): add a staff employee as a team member. To keep the system in a consistent state, we also need to update the staff to have the calling manager object as his/her manager. Follow this algorithm:

check whether the parameter object is already in the arraylist

If yes, return false // don’t add if already exist. Fail to add

// now okay to add

add the parameter staff to the arraylist field

set this object (the calling manager) as the manager of the parameter staff // Hint: xx.setManager(this)

return true // success

* removeTeamMember(): remove a staff employee from this calling manager’s team. To keep the system in a consistent state, we also need to update the staff to “remove” his/her manager. Follow this algorithm:

check whether the parameter object is already in the arraylist

If not, return false // can’t remove someone who’s not there

// now okay to remove

remove the parameter staff from the arraylist

set the manager of the parameter staff to null // Hint: xx.setManager(xxx)

return true // success

* toString(): of the calling object in this format:

*str-returned-by-toString() of its super*, Manages: [full name of staff 1 in the arraylist, full name of staff 2, ...]

Update the main() of your TestHr:

Create a staff obj #1

Create a staff obj #2

Create a Manager obj #1

Create a Manager obj #2

(add) call method to add staff #1 as a team member of manager #1

(add) call method to add manager #2 as a team member of manager #1

(add) call method to add staff #2 as a team member of manager #2

Print staff #1

// should show Manager: manager #1

Print staff #2

// should show Manager: manager #2

Print manager #1

// should show Manager: none, Manages: [staff #1, manager #2]

Print manager #2

// should show Manager: manager #1, Manages: [staff #2]

For your reference, here is the output of a sample run (yours should show different data):

ID: 101, Name: S1Bob Ellen, Position: Technician I, HiringDate: 2019-09-22, Salary: $80,000 (Bonus: $6,000), Manager: M1John Doe

ID: 105, Name: S2Kate Houston, Position: Technician II, HiringDate: 2020-09-22, Salary: $70,000 (Bonus: $5,250), Manager: M2Mary Carpenter

ID: 201, Name: M1John Doe, Position: Tech Lead II, HiringDate: 2017-09-22, Salary: $150,000 (Bonus: $15,000), Manager: none, Manages: [S1Bob Ellen, M2Mary Carpenter]

ID: 205, Name: M2Mary Carpenter, Position: Tech Lead I, HiringDate: 2019-09-22, Salary: $120,000 (Bonus: $12,000), Manager: M1John Doe, Manages: [S2Kate Houston]

Finally add those at the end of the main():

Call method to remove manager #2 from manager #1’s team

Print manager #1 again

Print manager #2 again

The additional prints will show both managers with “none” for their Manager fields and each has one single member on their team.

Take a screenshot of the final execution of your program and include it in the assignment report document.

1. Reflection: answer those questions AFTER you’ve completed this assignment:
2. What’s the hardest part of this assignment for you? Please explain.
3. Were you able to understand the relationships between classes in this assignment? How has your understanding of inheritance and composition changed by the end of this assignment?

## SUBMISSION

Submit four .java files + one word/PDF document. Please put screenshots and answers to questions into your word/PDF document.

* Exercise 1: one screenshot showing package setup.
* Exercise 2: a java file and answer to the question.
* Exercise 3-7: three java files and a required screenshot of execution.
* Exercise 8: Assignment reflection
* Check the completeness of your work against the rubric before turning it in.

## Rubric: Unit 5 Assignment

| **Criteria** | **Ratings** | | | **Pts** |
| --- | --- | --- | --- | --- |
| **Exercise 1 (package setup)** |  | 0.5 pts. Screenshot showing correct package setup. | 0 pts. Incorrect or no submission. | 0.5 |
| **Exercise 2**  **(Employee class)** |  | 0.5 pts. Correct response. | 0 pts. Incorrect or no submission | 0.5 |
| **Exercise 3-7 (Staff class)** | 5 pts. Correct class meeting all requirements. | 4.5 ~ 1 pts. At least one requirement is incorrect or missing:  (0.5 pts) extends the Employee class;  (0.5 pts) default constructor;  (1 pts) parameterized constructor;  (0.5 pts) getter;  (0.5 pts) setter;  (1 pts) getAnnualBonus();  (1 pts) toString(). | 0 pts. Incorrect or missed all requirements; no submission | 5 |
| **Exercise 3-7 (Manager class)** | 9 pts. Correct class meeting all requirements. | 8.5 ~ 1 pts. At least one requirement is incorrect or missing:  (0.5 pts) extends the Staff class;  (0.5 pts) default constructor;  (1 pts) parameterized constructor;  (2 pts) addTeamMember();  (2 pts) removeTeamMember();  (1 pts) getAnnualBonus();  (2 pts) toString(). | 0 pts. Incorrect or missed all requirements; no submission | 9 |
| **Exercise 3-7 (client class)** | 2 pts. Correct main() meeting all requirements. | 1.5 ~ 1 pts. At least one requirement is incorrect or missing:  (0.5 pts) create two different staff objects;  (0.5 pts) create two different manager objects;  (0.5 pts) call methods to add team members and remove.  (0.5 pts) print objects. | 0 pts. Incorrect or missed all requirements; no submission | 2 |
| **Exercise 8. Reflection** | 1 pts. Answered both questions. | 0.5 pts. Answered only one question. | 0 pts. Didn’t answer the questions or no submission. | 1 |
| **Required program name (StuPre part) + Style Points (Proper comments; meaning identifier names; consistent indentation)** | 2 pts. Correctly named program and proper style in all three areas of style points. | 1 pts. Problems in one of the following areas: program name, three areas of style points. | 0 pts. Problems in two or more of the following areas: program name, three areas of style points. | 2 |
|  |  |  | *Total Points* | 20 |