**San José State University**

# Computer Science Department CS151, Object Oriented Design and Programming, 07, Spring 2020

# Homework #2

Objective:

This homework’s objective is to review and understand the unit on classes and interfaces, when to use one or another, and various ways to implement them in Java.

Details:

Exercise 1:

Define and implement class **Student**. This class should contain the following fields: first name, last name, age, gpa, major, department. Age should be an integer value. GPA should be a floating point value. This class should contain getters and setters for all its attributes. This class also needs at least one constructor. Within class Student implement a nested non-static inner class called **Course**. This class should print out the student’s course schedule to command line via method called *printSchedule()*. This student’s schedule is up to your implementation. Save this class and its definition into a file named **Student.java**.

Define and implement class **StudentTest**.This class should implement *main()* method. In the body of the *main()* method you should create an instance of Student with the following information: John Smith, 20 year old, 3.6 gpa, Computer Science major, School of Computer Science department. You should make an appropriate call and print this student’s schedule. Save this class and its definition into a file named **StudentTest.java**.

Exercise 2:

List main differences between primitive data type **int** and class **Integer**. Be as specific as possible. What are advantages of using each one? List all specific functionalities that are available when using one and not the other. Consult Java API as a resource to find out this information. Save your answer to a text file named **exercise2.txt**.

Exercise 3:

Define and implement class **Person**. This class should contain the following fields: first name, last name, age, social security number, address, gender, and weight. Age should be an integer value, while weight should be a floating point value. This class should contain getters and setters for all its attributes. This class also needs at least one constructor. This class should also overload implementation of *toString()* method and implement new method named *introduce()*, which will display this person’s attribute values to the command line. Save this class and its definition into a file named **Person.java**.

Define and implement class **Employee**. An employee is a person, so has all the attributes and functionality of a person. In addition to being a person, an employee should have the following attributes: employee id, employee status (contractor, full time, part time), start date, hourly pay (for part time employees and contractors) or yearly salary (for full time employees). This class should contain getters and setters for all its attributes. In addition to all the functionality of a person, employee should be able to introduce themselves. This means they need to be able to display to command line all their information as a person as well as additional employee information. Employees should also be able to compute pay. Those employees that get paid hourly need a number of work hours specified. Salaried employees should not require the number of hours to be specified, but rather number of weeks they are getting paid for. You have full creative license on how you choose to implement this functionality (method overloading or a single function that checks the particular employee’s status and interprets the input accordingly). Remember that if you choose to implement using method overloading you still need to check that the correct version of the method is being called on the correct employee status. You should call this method *calculatePay()*. During the object instantiation, the constructor of this class should pass on instantiation of superclass’ attributes to the superclass constructor. Save this class and its definition into a file named **Employee.java**.

Define and implement class **EmployeeTest**.This class should implement *main()* method. In the body of the *main()* method you should create the following Employee instances, call *introduce()* and *calculatePay()* on each one and display the results of *calculatePay()* method to command line:

Joe Smith, a contractor, pay is $60/hr, should get paid or working 30 hours, other details are up to you

Lisa Gray, a full time employee, pay is $110,000/yr, should get paid or working 2 weeks, other details are up to you

Timothy Briggs, a full time employee, pay is $80,000/yr, should get paid or working 4 weeks, other details are up to you

George Wallace, a part time employee, pay is $20/hr, should get paid or working 25 hours, other details are up to you

Amy Student, a contractor employee, pay is $45/hr, should get paid or working 45 hours, other details are up to you

For visual presentation make sure to include an empty line between each employee instance output. Save this class and its definition into a file named **EmployeeTest.java**.

Exercise 4:

Define and implement an immutable class **Product**. This class should contain the following fields: product name, product description, product price, maximum quantity allowed to be ordered. This class should contain getters for all its attributes. This class also needs at least one constructor. This class should also overload implementation of *toString()* method. Save this class and its definition into a file named **Product.java**.

Define and implement class **ProductTest**.This class should implement *main()* method. In the body of the *main()* method you should create an instance of Product (details of the product are up to you). Save this class and its definition into a file named **ProductTest.java**.

Exercise 5:

This exercise describes what the expected behavior and functionality are. The specific details of implementation and decisions about where to use abstract or concrete classes or interfaces are up to you. You have full creative license in how you design and implement your program. The only requirement for your design is that you should demonstrate both class and interface inheritance by the same class.

Define and implement the following classes **Dog**, **Cat**, **Racoon**, **Whale**. Save the class definitions into files named **Dog.java**, **Cat.java**, **Racoon.java**, **Whale.java**. Each of these classes should define or inherit the following attributes: type of the animal, name of the animal, age, male or female, which environment it lives in (home for dogs and cats, forest for racoons, ocean for whales), and how fast it moves (e.g. x miles/hr). Each of these classes should contain getters and setters for all their attributes. Each of these classes should have the following functionality where the action is simply printed to command line: move(), sound(), eat(), sleep(), toString(). All classes are a type of animal. Some of the classes are domesticated (for example, you could introduce Domesticated interface to your design). Being domesticated introduces additional functionality: walk(), greetHuman(). Some of the classes are swimmers, which adds functionality swim(). Dog has an additional functionality bark(). Both Racoon and Cat can scratch().

Define and implement class **AnimalTest**.This class should implement *main()* method. In the body of the *main()* method you should create one instance of your classes Dog, Cat, Racoon, and Whale. On each instance call each one of the possible functionalities (methods representing those functionalities). These calls should just output the appropriate action to command line. For example, calling method walk() should output something like “Walking”. For visual presentation make sure to include an empty line between the outputs for each animal instance. Save this class and its definition into a file named **AnimalTest.java**.

Submission:

In your class repo create a directory called “Assignment2” and add all the files created for this homework assignment to that directory.

This homework assignment is assigned on 02/04/2020 and is due on 02/18/2020 before 11:59pm. Email your assignment submission to me at both [Yulia.Newton@sjsu.edu](mailto:Yulia.Newton@sjsu.edu) and [yulia.newton@gmail.com](mailto:yulia.newton@gmail.com), as well as the grader at [akshay.kajale@sjsu.edu](mailto:akshay.kajale@sjsu.edu). The subject of the email should say “CS151 Assignment 2”. Add your name as it appears on the class roster and the URL to your git repo in the body of an email.

Grading:

Your code must compile and execute successfully in order to get full credit for this assignment. For each exercise, except exercise # 2, I will compile and execute xxxTestxxx.java file listed in that exercise description.

* Program with no compile errors (5 pts)
* Program executes (5 pts)
* Program outputs what is required by the exercise (5 pts)
* A total of 60 points are possible for combined exercises #1, 3, 4, 5

Exercise #2 will be graded based on my reading the answer (5 pts if the answer is correct and complete; 2.5 pts if the answer is correct but incomplete; 0 pts if the answer is incorrect).

A total of 65 points are possible for this homework assignment.