CSCI 218: Programming II (Spring 2025)

Practice Exercise

Due Date: January 30th 2025, 12:30pm

Goal: The objective of these class exercises is to assist you in grasping the basics of Java and developing confidence in writing Java code. A starter file accompanied by comments to support and guide you throughout the coding process has been provided. Please don't hesitate to get my attention at any time during the class.

Task 1: Loops

Use for loop to get the factorials of n positive integers 1,2,…,1000

Task 2: Loops

Write a java program to get the sum of the series

Sum= $1-\frac{1}{2}+\frac{1}{3}-\frac{1}{4}+\ldots+\frac{1}{99}-\frac{1}{100}$

Task 3: Loops

Create a java method to get the sum of a square of odd numbers between 22 and 389

Task 4: OOP - Animal

Write a class called Animal that has the following members:

- An instance field of type float called mass.
- An instance field of type String called name for the species.
- An instance field of type int called legs.
- An instance field of type boolean called isAlive.
- A class field of type int called counter to keep track of the total number of animals birthed.
- A class field of type int called animals_alive to keep track of the number of animals alive.
- A setter method for the field mass.
- Getter methods for the three fields.
- A constructor that takes one argument of type String and uses it to initialize the field name and sets isAlive to true. It should increase the counter by 1 every time an object is created. The other two fields should be set to 0.
- Another constructor takes three arguments of type float, String and int and initializes the three fields. It should set isAlive to true. It should increase the counter by 1 every time an object is created.

- A method **print** that outputs a String in the form "animal (name =goat, mass=15.2, legs=4, status='Alive/Dead')".
- A method **getAnimalPopulation** that returns the total number of animals who have ever lived.
- A method **getAnimalsAlive** that returns the total number of animals Alive.
- A method **reproduce()** that returns a new Animal object with the same species, number of legs and half the mass of its parent.
- A method **die()** that decreases animals_alive by one and sets the isAlive to false
- Write a class called **TestAnimal** with a main method
    1. Creates an Animal object using the three-argument constructor, initializing it to name " snail " , mass 3.5, and legs 1, and then prints out the object using the print method.
    2. Display a message showing the world population count and number of animal alive
    3. Create another animal Object using the one-argument constructor and then prints out the object using the print method.
    4. Repeat Step 2
    5. Change the mass of the animal object created in Step 3 to 4.2
    6. Creates another Animal object of your choice using the three-argument constructor. Print out the object using the print method. Repeat Step 2
    7. Declare 2 Animal objects; babyAnimal1, babyAnimal2
    8. Call the reproduce() method on your animal objects created in Step 3 and 6 and assign them to the objects declared in Step 7
    9. Call the die() method on the animal object created in Step 1. Repeat step 2
    10. Using the babyAnimal1 object, access the counter variable and change its value to 500
    11. Using any of the 3 animal objects created, perform step 2 again
    12. Use the Animal class name, access the counter variable and change its value to 15. Repeat Step 11.
    13. Call the die() method on all the animal objects created
    14. Perform step 2

Task 5: OOP - ChangeValue

Write a Java class called **ChangeValue** that declares 2 integer instance variables: value1 and value2. These should be declared private.

- Your class should have a constructor that initializes the two variables.
- In the class provide a **method print()** that displays the result of multiplying value1 and value2.
- Create a Driver Class for your program that has a Main Method. In your main method, declare two integer variables: val1 and val2.
- Use the Scanner class to take the two integer entries from the keyboard and pass them to the **ChangeValue** constructor.

- In the **ChangeValue** constructor evaluate the two input values that were passed using if statements:
    - if val1 is greater than 5 then set instance variable value1 equal to the first value passed to the constructor.
    - if val1 is less than or equal to 5 then set instance variable value1 equal to the first passed value plus the second passed value.
    - if val2 is less than 10 then set instance variable value2 equal to the second passed value times the second passed value and then add 5.
    - if val2 is greater to or equal to 10 then set instance variable value2 equal to the second passed value.
- From Main call the **print** method which should print the resulting computed values in the class object.

The output might look like this:

```
Enter an integer value: 8
Enter a second integer value: 3

The calculated value is: 112

Press any key to continue . . .
```

Task 6: OOP - Modelling a House

Define a class named House that stores information about a house. It should comprise the following:

- Private instance variables to store age of the House, its type (Detached, Semi-Attached, Attached) and its cost.
- 4 constructors: No argument (sets age to 50, type to Attached and cost to 100000), one argument constructor (sets cost to a value, age - 50 and type - Attached), two argument constructors (sets age to a value, cost to a value , and type to Attached), three argument constructors (sets age to a value, cost to a value, and type to Attached, semi-detached, or detached)
- 3 Accessor/Getter methods: - methods to return age, type and cost respectively
- 5 Mutator/Setter methods: - 3 methods for setting the three values independently, a method to set all three values and a method to set age and cost of the house.
- A public method called estimatePrice() that returns cost of a house based on type and age. An attached cost $100,000, appreciates 1% every year in first five years and 2% every year afterwards. A Semidetached cost $150,000, appreciates 2% every year in first five years and 3% every year afterwards. A detached costs $200,000, appreciates 2% every year in first five years and 2% every year afterwards.
- A toString() method that returns type of the house and its age and its cost.

- An equals() method to test for equality of two objects of class House based on type and age.
- isLessThan() and isGreaterThan() method to compare between the prices of two objects of class House.

Write a test code

- Which declares 4 house objects using 4 different constructors and output description of the 4 houses.
- Test your accessor methods.
- Calculate the estimated price of houses given type and age (include 1 attached and 1 detached)
- Test out all 5 mutator methods to modify the attributes of different House objects.
- Test methods toString(), equals() , isLessThan() and isGreaterThan() for different House objects.

Here is an example of the output to illustrate the expected behavior of your program.

```
House H1: This House is type attached. Its age is 1950 and costs $100000.0
House H2: This House is type attached. Its age is 1950 and costs $100000.0
House H3: This House is type attached. Its age is 4 and costs $120000.0
House H4: This House is type detached. Its age is 2 and costs $220000.0

Accessor Method: The housetype for house H4 is detached, its age is 2, and it costs $220000.0

The estimated price of house H3 is $104800.0
The estimated price of car H4 is $208800.0

Mutator Method: The new age for house H3 is 5
Mutator Method: The new housetype for house H3 is semi-detached
Mutator Method: The new cost for house H3 is 240000.0
Mutator Method: The new house H3 age is 6 and its new cost is 245000.0
Mutator Method: The new housetype for house H3 is semi-detached, its new age is 14and its
cost is 260000.0

toString: This House is type semi-detached. Its age is 14 and costs $260000.0

Houses H1 and H2 are equal is true
Houses H1 and H4 are equal is false

House H4 is less than H3 is true

House H1 is greater than H3 is false
```