Session 9: Programming Exercises

1. **Pet Class**

   Write a class named Pet, which should have the following data attributes:

   - _ _name (for the name of a pet)
   - _ _animal_type (for the type of animal that a pet is. Example values are 'Dog', 'Cat', and 'Bird')
   - _ _age (for the pet's age)
   - The Pet class should have an _ _init_ _ method that creates these attributes. It should also have the following methods:
     - set_name
     - This method assigns a value to the _ _name field.
     - set_animal_type
     - This method assigns a value to the _ _animal_type field.
     - set_age
     - This method assigns a value to the _ _age field.
     - get_name
     - This method returns the value of the _ _ name field.
     - get_animal_type
     - This method returns the value of the _ _animal_type field.
     - get_age
     - This method returns the value of the _ _age field.
   - Once you have written the class, write a program that creates an object of the class and prompts the user to enter the name, type, and age of his or her pet. This data should be stored as the object's attributes. Use the object's accessor methods to retrieve the pet's name, type, and age and display this data on the screen.

2. **Car Class**

   Write a class named Car that has the following data attributes:

   - _ _year_model (for the car's year model)
   - _ _make (for the make of the car)
   - _ _speed (for the car's current speed)

   - The Car class should have an _ _init_ _ method that accepts the car's year model and make as arguments. These values should be assigned to the object's _ _year_model and _ _make data attributes. It should also assign 0 to the _ _speed data attribute.

   - The class should also have the following methods:

     - accelerate

     - The accelerate method should add 5 to the speed data attribute each time it is called.

     - brake

     - The brake method should subtract 5 from the speed data attribute each time it is called.

     - get_speed

     - The get_speed method should return the current speed.

   - Next, design a program that creates a Car object then calls the accelerate method five times. After each call to the accelerate method, get the current speed of the car and display it. Then call the brake method five times. After each call to the brake method, get the current speed of the car and display it.

3. **Personal Information Class**

   Design a class that holds the following personal data: name, address, age, and phone number. Write appropriate accessor and mutator methods. Also, write a program that creates three instances of the class. One instance should hold your information, and the other two should hold your friends' or family members' information.

4. **Employee Class**

   Write a class named Employee that holds the following data about an employee in attributes: name, ID number, department, and job title.

   Once you have written the class, write a program that creates three Employee objects to hold the following data:

Name	ID Number	Department	Job Title
Susan Meyers	47899	Accounting	Vice President
Mark Jones	39119	IT	Programmer
Joy Rogers	81774	Manufacturing	Engineer

The program should store this data in the three objects, then display the data for each employee on the screen.

5. **RetailItem Class**

Write a class named RetailItem that holds data about an item in a retail store. The class should store the following data in attributes: item description, units in inventory, and price.

Once you have written the class, write a program that creates three RetailItem objects and stores the following data in them:

	Description	Units in Inventory	Price
Item #1	Jacket	12	59.95
Item #2	Designer Jeans	40	34.95
Item #3	Shirt	20	24.95

6. **Patient Charges**

Write a class named Patient that has attributes for the following data:

- First name, middle name, and last name
- Address, city, state, and ZIP code
- Phone number
- Name and phone number of emergency contact
- The Patient class's _ _init_ _ method should accept an argument for each attribute. The Patient class should also have accessor and mutator methods for each attribute.
- Next, write a class named Procedure that represents a medical procedure that has been performed on a patient. The Procedure class should have attributes for the following data:
    - Name of the procedure
    - Date of the procedure
    - Name of the practitioner who performed the procedure
    - Charges for the procedure

- The Procedure class's _ _init_ _ method should accept an argument for each attribute. The Procedure class should also have accessor and mutator methods for each attribute.

- Next, write a program that creates an instance of the Patient class, initialized with sample data. Then, create three instances of the Procedure class, initialized with the following data:

Procedure #1:	Procedure #2:	Procedure #3:
Procedure name: Physical Exam Date: Today's date Practitioner: Dr. Irvine Charge: 250.00	Procedure name: X-ray Date: Today's date Practitioner: Dr. Jamison Charge: 500.00	Procedure name: Blood test Date: Today's date Practitioner: Dr. Smith Charge: 200.00

- The program should display the patient's information, information about all three of the procedures, and the total charges of the three procedures.

7. Employee Management System

This exercise assumes you have created the Employee class for Programming Exercise 4. Create a program that stores Employee objects in a dictionary. Use the employee ID number as the key. The program should present a menu that lets the user perform the following actions:

- Look up an employee in the dictionary
- Add a new employee to the dictionary
- Change an existing employee's name, department, and job title in the dictionary
- Delete an employee from the dictionary
- Quit the program

- When the program ends, it should pickle the dictionary and save it to a file. Each time the program starts, it should try to load the pickled dictionary from the file. If the file does not exist, the program should start with an empty dictionary.

8. Cash Register

This exercise assumes you have created the RetailItem class for Programming Exercise 5. Create a CashRegister class that can be used with the RetailItem class. The CashRegister class should be able to internally keep a list of RetailItem objects. The class should have the following methods:

- A method named purchase_item that accepts a RetailItem object as an argument. Each time the purchase_item method is called, the RetailItem object that is passed as an argument should be added to the list.
- A method named get_total that returns the total price of all the RetailItem objects stored in the CashRegister object's internal list.
- A method named show_items that displays data about the RetailItem objects stored in the CashRegister object's internal list.
- A method named clear that should clear the CashRegister object's internal list.

- Demonstrate the CashRegister class in a program that allows the user to select several items for purchase. When the user is ready to check out, the program should display a list of all the items he or she has selected for purchase, as well as the total price.

9. Trivia Game

In this programming exercise, you will create a simple trivia game for two players. The program will work like this:

  - Starting with player 1, each player gets a turn at answering 5 trivia questions. (There should be a total of 10 questions.) When a question is displayed, 4 possible answers are also displayed. Only one of the answers is correct, and if the player selects the correct answer, he or she earns a point.
  - After answers have been selected for all the questions, the program displays the number of points earned by each player and declares the player with the highest number of points the winner.

- To create this program, write a Question class to hold the data for a trivia question. The Question class should have attributes for the following data:

  - A trivia question
  - Possible answer 1
  - Possible answer 2
  - Possible answer 3
  - Possible answer 4
  - The number of the correct answer (1, 2, 3, or 4)

- The Question class also should have an appropriate _ _init_ _ method, accessors, and mutators.

- The program should have a list or a dictionary containing 10 Question objects, one for each trivia question. Make up your own trivia questions on the subject or subjects of your choice for the objects.

10. Employee and ProductionWorker Classes

Write an Employee class that keeps data attributes for the following pieces of information:

- Employee name
- Employee number

- Next, write a class named ProductionWorker that is a subclass of the Employee class. The ProductionWorker class should keep data attributes for the following information:
  - Shift number (an integer, such as 1, 2, or 3)
  - Hourly pay rate
- The workday is divided into two shifts: day and night. The shift attribute will hold an integer value representing the shift that the employee works. The day shift is shift 1 and the night shift is shift 2. Write the appropriate accessor and mutator methods for each class.
- Once you have written the classes, write a program that creates an object of the ProductionWorker class and prompts the user to enter data for each of the object's data attributes. Store the data in the object, then use the object's accessor methods to retrieve it and display it on the screen.

11. ShiftSupervisor Class

In a particular factory, a shift supervisor is a salaried employee who supervises a shift. In addition to a salary, the shift supervisor earns a yearly bonus when his or her shift meets production goals. Write a ShiftSupervisor class that is a subclass of the Employee class you created in Programming Exercise 1. The ShiftSupervisor class should keep a data attribute for the annual salary, and a data attribute for the annual production bonus that a shift supervisor has earned. Demonstrate the class by writing a program that uses a ShiftSupervisor object.

12. Person and Customer Classes

The Person and Customer Classes

Write a class named Person with data attributes for a person's name, address, and telephone number. Next, write a class named Customer that is a subclass of the Person class. The Customer class should have a data attribute for a customer number, and a Boolean data attribute indicating whether the customer wishes to be on a mailing list. Demonstrate an instance of the Customer class in a simple program.