Workshop #3: Class and Object

Learning Outcomes:

Upon successful completion of this workshop, you will have demonstrated the abilities to:

- Design and implement a class.
- Create an object from a class
- Describe to your instructor what you have learned in completing this workshop.

Requirements:

Part 1: [2 points]
To complete this task you should read and study the lecture Encapsulation
-Create a new project named "**CarManager**". It contains the file **Car.java and Tester.java.**
-In the file Car.java, you implement the Car class base on the class diagram as below:

Car
-Colour: String **-EnginePower:int** **-Convertible: boolean** **-ParkingBrake: boolean**
//constructors **+Car()** **+Car(String Colour, int EnginePower, boolean Convertible, boolean ParkingBrake)** **//getters** **+getColour():String** **+getEnginePower():int** **+getConvertible(): boolean** **+getParkingBrake(): boolean** **//setters** **+setColour(String colour):void** **+setEnginePower(int EnginePower):void** **+setConvertible(boolean Convertible): void** **+setParkingBrake(boolean ParkingBrake): void** **//other logic methods** **+pressStartButton():void** **+pressAcceleratorButton():void** **+output(): void**

- Default constructor: to assign all fields to empty values
- Constructor with parameters: to assign all fields to input parameters
- Getters: to return the value of a field
- Setters: to change the value of a field
- The method pressStartButton(): print out the message "You have pressed the start button"

- The method pressAcceleratorButton(): print out the message “You have pressed the Accelerator button”
- The method output(): print out values of all fields

- In the file “Test.java”. you type like as follow:

```
public class Tester {
        public static void main(String[] args) {
          Car c=new Car();
          c.pressStartButton();
          c.pressAcceleratorButton();
          c.output();


          Car c2=new Car("red", 100, true, true);
          c2.pressAcceleratorButton();
          c2.setColour("black");
          System.out.println("Colour of c2:" + c2.getColour());
          c2.output();
    }
}
```

- Run the method main to see the output

Part 2: Find classes [3 points] and use UML to draw class structure (optional)

Problem: Mr. Hung is the owner of the shop that sells guitars. He wants you to build him a shop management app. This app is used for keeping track of guitars. Each guitar contains serialNumber, price, builder, model, back Wood, top Wood. The guitar can create a melodious sound. The shop will keep guitars in the inventory. The owner can add a new guitar to it, he search also the guitar by the serialNumber.

Requirement: Write your paper down classes in the problem and use UML draw class structure.

Note: show members of a class: fields and methods

Do it yourself before getting help

Guideline**:**

Apply design guideline :

- Main nouns: Guitar
- Auxiliary nouns describe details of the guitar:serialNumber, price, builder, model, back Wood, top Wood.
- Verbs describe behaviors of the guitar: create Sound

Continue finding other nouns
- Main nouns: Inventory
- Auxiliary nouns describe details of the inventory: the list(array) of guitars
- Verbs describe behaviors of the inventory: add a new guitar, search the guitar by serialNumber.

Using UML to draw a class diagram

Part 3: only implement the Guitar class [4 points].

Step by step workshop instructions:

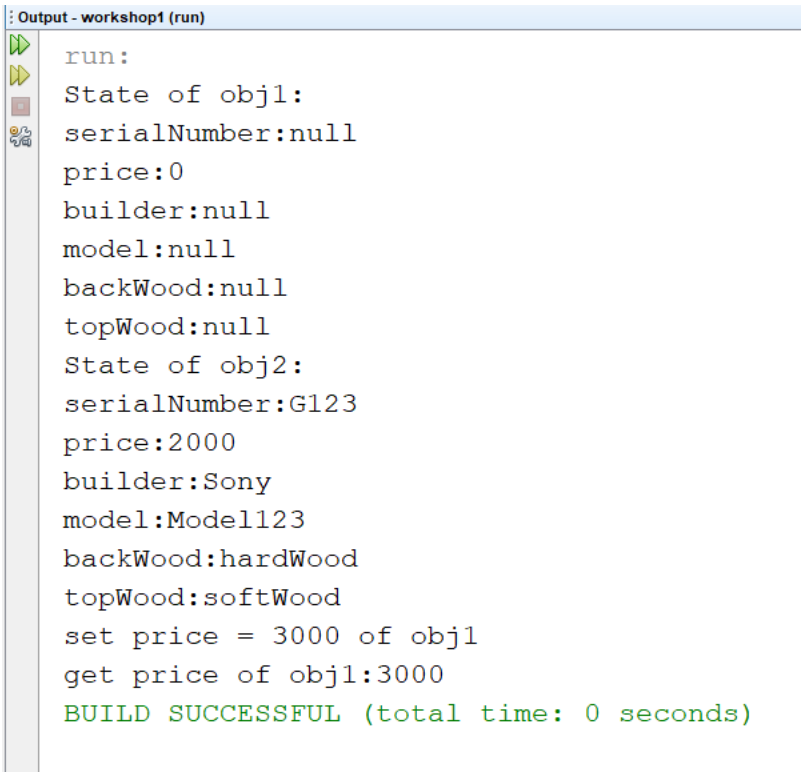
- Create a new project named "**workshop3**"
- In the project, create a new file named "**Guitar.java"**
    - Declare fields with access modifier as private: String serialNumber, int price, String builder, String model, String backWood, String topWood
    - Declare and implement methods with access modifier as public:
        - public Guitar() {…} : to assign all fields to empty values
        - public Guitar(String serialNumber, int price, String builder, String model, String backWood, String topWood) {…}: to assign all fields by input parameters
        - public String getSerialNumber(){…}: return the value of the field serialNumber.
        - public void setSerialNumber(String serialNumber){…}: if the input parameter is not empty then assign it to the field serialNumber.
        - Implement getter/setter of all other fields
        - public void createSound(){…}: in the method, invoke all getters and use System.out to print out values after getting.

- In the project, create a new file named "**Tester.java.** Create the method main in here, you type:

```
public class Tester {
      public static void main(String[] args) {
       Guitar obj1=new Guitar();
       Guitar obj2=new Guitar("G123",2000,"Sony","Model123","hardWood","softWood");
       System.out.println("State of obj1:");
       obj1.createSound();
       System.out.println("State of obj2:");
       obj2.createSound();

       System.out.println("set price = 3000 of obj1");
       obj1.setPrice(3000);
       System.out.println("get price of obj1:" + obj1.getPrice() );
   }
}
```

The output is:



Part 4: Draw the memory map when the program runs [1 point]

Explain step by step what happened when the program runs and answer some questions.

- What is stored in the static heap, stack, dynamic heap?
- What are objects in the program?
- What is the state of obj1, obj2?
- Do you access all fields of obj1 in the class Tester.java? Why?
- What is the current object when the program runs to the line "obj2.createSound();"?
- In the method main, can you use the keyword "this" to access all fields of obj2? Why?