Workshop #3: Class and Object

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.

To complete this task you should read and study the lecture Encapsulation.

1) 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

Where:

- 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.

2) **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. Let's implement the Guitar class.**

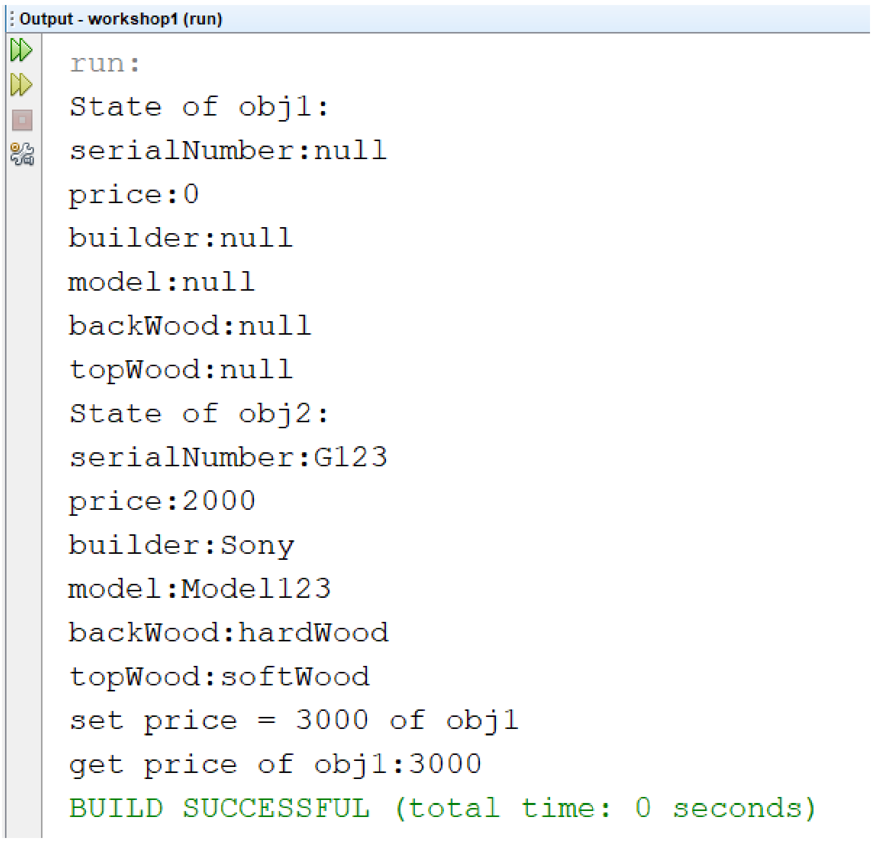
Step by step workshop instructions:

- Create a new project named "**GuitarManager**"
- 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:

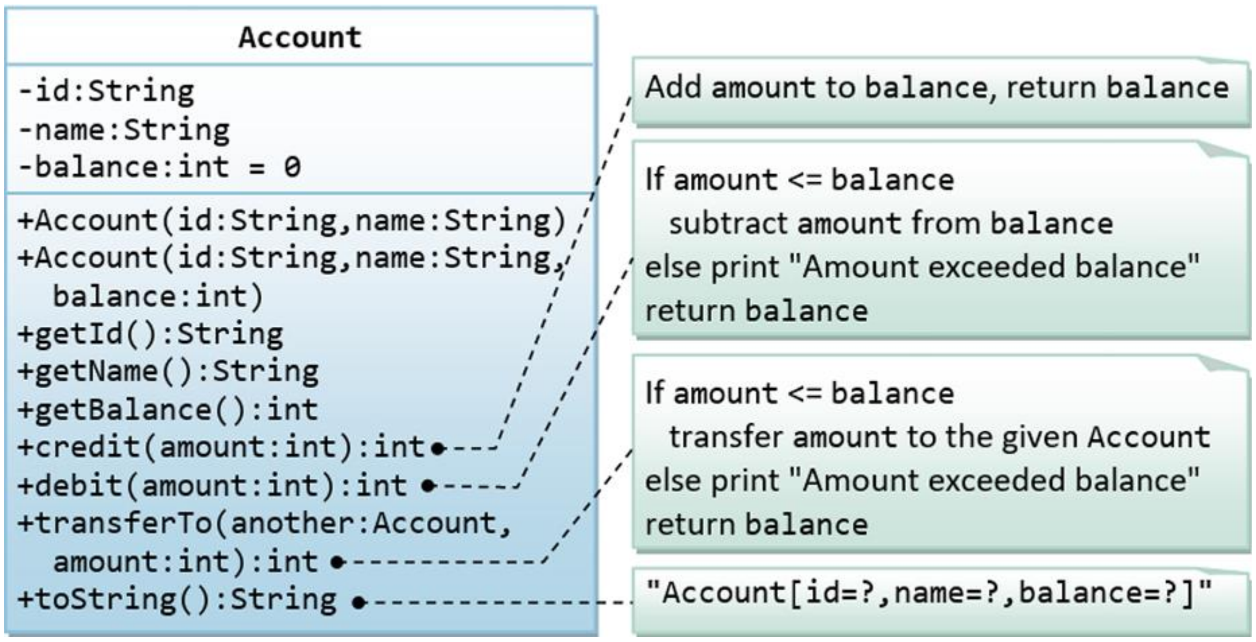


```
run:
State of obj1:
serialNumber:null
price:0
builder:null
model:null
backWood:null
topWood:null
State of obj2:
serialNumber:G123
price:2000
builder:Sony
model:Model123
backWood:hardWood
topWood:softWood
set price = 3000 of obj1
get price of obj1:3000
BUILD SUCCESSFUL (total time: 0 seconds)
```

3) **A class called Account, which models a bank account of a customer, is designed as shown in the following class diagram.**

The methods credit(amount) and debit(amount) add or subtract the given amount to the balance. The method transferTo(anotherAccount, amount) transfers the given amount from this Account to the given anotherAccount.

Let's write the Account class.


Account
-id:String
-name:String
-balance:int = 0
+Account(id:String,name:String)
+Account(id:String,name:String, balance:int)
+getId():String
+getName():String
+getBalance():int
+credit(amount:int):int
+debit(amount:int):int
+transferTo(another:Account, amount:int):int
+toString():String
Add amount to balance, return balance
If amount <= balance subtract amount from balance else print "Amount exceeded balance" return balance
If amount <= balance transfer amount to the given Account else print "Amount exceeded balance" return balance
"Account[id=?,name=?,balance=?]"

Below is a test driver to test the Account class:

```
public class TestMain {
   public static void main(String[] args) {
      // Test constructor and toString()
      Account a1 = new Account("A101", "Tan Ah Teck", 88);
      System.out.println(a1);  // toString();
      Account a2 = new Account("A102", "Kumar"); // default balance
      System.out.println(a2);

      // Test Getters
      System.out.println("ID: " + a1.getID()); System.out.println("Name: "
      + a1.getName()); System.out.println("Balance: " + a1.getBalance());

      // Test credit() and debit()
      a1.credit(100);
      System.out.println(a1);
      a1.debit(50);
      System.out.println(a1);
      a1.debit(500);  // debit() error
      System.out.println(a1);

      // Test transfer()
      a1.transferTo(a2, 100);  // toString()
      System.out.println(a1);
      System.out.println(a2);
   }
}
```

The expected output is:

```
Account[id=A101,name=Tan Ah Teck,balance=88]
Account[id=A102,name=Kumar,balance=0]
ID: A101
Name: Tan Ah Teck
Balance: 88
Account[id=A101,name=Tan Ah Teck,balance=188]
Account[id=A101,name=Tan Ah Teck,balance=138]
Amount exceeded balance
Account[id=A101,name=Tan Ah Teck,balance=138]
Account[id=A101,name=Tan Ah Teck,balance=38]
Account[id=A102,name=Kumar,balance=100]
```

4) Write a class Item (in the default package of the NetBean) with the following information:

Item
-name:String
-quantity:int
+Item()
+Item(name:String, quantity:int)
+getName():String
+setName(name:String):void
+getQuantity():int
+setQuantity(quantity:int):void

Where:

- Item() - default constructor.
- Item(name:String, quantity:int) - constructor, which sets values to name and quantity.
- getName():String - return name in **uppercase** format.
- setName(name:String):void - update name.
- getQuantity():int - return quantity.
- setQuantity(quantity:int):void - update quantity.

The program output might look something like:

```
Enter name: TiVi
Enter quantity: 8
1. Test getName()
2. Test setQuantity()
Enter TC (1 or 2): 1
OUTPUT:
TIVI
```

```
Enter name: TiVi
Enter quantity: 8
1. Test getName()
2. Test setQuantity()
Enter TC (1 or 2): 2
Enter new quantity: 12
OUTPUT:
12
```

5) Create a class called Employee that includes three pieces of information as instance variables: a first name (type String), a last name (type String), and a monthly salary (type double).

Your class should have a constructor that initializes the three instance variables. Provide a set and a get method for each instance variable. If the monthly salary is not positive, set it to 0.0.

Write a test application named EmployeeTest that demonstrates class Employee's capabilities. Create two Employee objects and display each object's monthly salary. Then give each Employee a 10% raise and display each Employee's monthly salary again.