Practical no. 3

FS19CO042

Aim:

3.1 Define the following classes/ interfaces with the help of above shortcuts:

1. Person(id, name, dateOfBirth, age, street, city, pin : default and parameterized constructors and setters and getters)
2. Department(id, name, dateOfEstablishment, headOfficeLocation, headId, numberOfEmployees : default and parameterized constructors and setters and getters)
3. Point(x, y, z : default and parameterized constructors and setters and getters)
4. Vehicle(registrationNumber, rcBookNumber, manufacturer, numberOfWheels, vehicleType, model, numberOfSeats : default and parameterized constructors and setters and getters)
5. Laptop(imeiNumber, processorName, processorSpeed, primaryMemoryType, primaryMemoryCapacity, secondaryStorageType, secondaryStorageCapaciry, screenResolution, screenType, isLED, listOfPorts, osInstalled : default and parameterized constructors and setters and getters)
6. interface Taxable(public int cost(), public intpercentGST())

3.2 Check whether feature of Encapsulation has been followed in 3.1. If not make necessary changes.

3.3 Define classes Car, Train and Truck with necessary member fields, constructors and methods. Make them extend class Vehicle.

3.4 Define a class Gadget with necessary member fields, constrictors and methods. Modify the class Laptop to extend the class Gadget.

3.5 In main method, declare a reference variable vehicle of class Vehicle and create an object of class Car which will be referenced by vehicle. Call getName() method on the object. (Hint: Reference Variable Casting)

3.6 Modify the classes Vehicle and Gadget implement the interface Taxable. Hence override respective methods.

3.7 Modify the classes Car and Laptop to override the implemented methods in 3.6.

3.8 Modify the class Gadget to add a data member gadgetCount such that its value will incremented as soon as a new object is initialized. Create 5 objects of the class Print its value after initializing each object.

Tool used: Editor (Notepad/Intellij IDE), JDK and JRE

Theory:

Java is an Object-Oriented Language. As a language that has the Object-Oriented feature, Java supports the following fundamental concepts −

- Polymorphism
- Inheritance
- Encapsulation
- Abstraction
- Classes
- Objects

- Instance
- Method
- Message Passing

Object − Objects have states and behaviors. Example: A dog has states - color, name, breed as well as behaviors – wagging the tail, barking, eating. An object is an instance of a class.

Class − A class can be defined as a template/blueprint that describes the behavior/state that the object of its type support.

Objects in Java

Let us now look deep into what are objects. If we consider the real-world, we can find many objects around us, cars, dogs, humans, etc. All these objects have a state and a behavior.

If we consider a dog, then its state is - name, breed, color, and the behavior is - barking, wagging the tail, running.

If you compare the software object with a real-world object, they have very similar characteristics.

Software objects also have a state and a behavior. A software object's state is stored in fields and behavior is shown via methods.

So in software development, methods operate on the internal state of an object and the object-to-object communication is done via methods.

Classes in Java

A class is a blueprint from which individual objects are created.

Following is a sample of a class.

Example

```
public class Dog {
   String breed;
   int age;
   String color;
   void barking() {
   }
   void hungry() {
   }
   void sleeping() {
   }
}
```

A class can contain any of the following variable types.

- **Local variables −** Variables defined inside methods, constructors or blocks are called local variables. The variable will be declared and initialized within the method and the variable will be destroyed when the method has completed.
- **Instance variables −** Instance variables are variables within a class but outside any method. These variables are initialized when the class is instantiated. Instance variables can be accessed from inside any method, constructor or blocks of that particular class.
- **Class variables −** Class variables are variables declared within a class, outside any method, with the static keyword.

A class can have any number of methods to access the value of various kinds of methods. In the above example, barking(), hungry() and sleeping() are methods.

Following are some of the important topics that need to be discussed when looking into classes of the Java Language.

Constructors

When discussing about classes, one of the most important sub topic would be constructors. Every class has a constructor. If we do not explicitly write a constructor for a class, the Java compiler builds a default constructor for that class.

Each time a new object is created, at least one constructor will be invoked. The main rule of constructors is that they should have the same name as the class. A class can have more than one constructor.

Following is an example of a constructor −

Example

```
public class Puppy {
   public Puppy() {
   }

   public Puppy(String name) {
      // This constructor has one parameter, name.
   }
}
```

Java also supports Singleton Classes where you would be able to create only one instance of a class.

Note − We have two different types of constructors. We are going to discuss constructors in detail in the subsequent chapters.

Creating an Object

As mentioned previously, a class provides the blueprints for objects. So basically, an object is created from a class. In Java, the new keyword is used to create new objects.

There are three steps when creating an object from a class −

- **Declaration** − A variable declaration with a variable name with an object type.
- **Instantiation** − The 'new' keyword is used to create the object.
- **Initialization** − The 'new' keyword is followed by a call to a constructor. This call initializes the new object.

An interface is a reference type in Java. It is similar to class. It is a collection of abstract methods. A class implements an interface, thereby inheriting the abstract methods of the interface.

Along with abstract methods, an interface may also contain constants, default methods, static methods, and nested types. Method bodies exist only for default methods and static methods.

Writing an interface is similar to writing a class. But a class describes the attributes and behaviors of an object. And an interface contains behaviors that a class implements.

Unless the class that implements the interface is abstract, all the methods of the interface need to be defined in the class.

An interface is similar to a class in the following ways −

- An interface can contain any number of methods.
- An interface is written in a file with a **.java** extension, with the name of the interface matching the name of the file.
- The byte code of an interface appears in a **.class** file.
- Interfaces appear in packages, and their corresponding bytecode file must be in a directory structure that matches the package name.

However, an interface is different from a class in several ways, including −

- You cannot instantiate an interface.
- An interface does not contain any constructors.
- All of the methods in an interface are abstract.
- An interface cannot contain instance fields. The only fields that can appear in an interface must be declared both static and final.
- An interface is not extended by a class; it is implemented by a class.
- An interface can extend multiple interfaces.

Declaring Interfaces

The **interface** keyword is used to declare an interface. Here is a simple example to declare an interface −

Example

Following is an example of an interface −

```
/* File name : NameOfInterface.java */

import java.lang.*;

// Any number of import statements

public interface NameOfInterface {

   // Any number of final, static fields

   // Any number of abstract method declarations\

}Interfaces have the following properties −
```

- An interface is implicitly abstract. You do not need to use the **abstract** keyword while declaring an interface.
- Each method in an interface is also implicitly abstract, so the abstract keyword is not needed.
- Methods in an interface are implicitly public.

Example

```
/* File name : Animal.java */
interface Animal {
   public void eat();
   public void travel();
}
```

Implementing Interfaces

When a class implements an interface, you can think of the class as signing a contract, agreeing to perform the specific behaviors of the interface. If a class does not perform all the behaviors of the interface, the class must declare itself as abstract.

A class uses the **implements** keyword to implement an interface. The implements keyword appears in the class declaration following the extends portion of the declaration.

Example

```
/* File name : MammalInt.java */
public class MammalInt implements Animal {
   public void eat() {
      System.out.println("Mammal eats");
   }
   public void travel() {
      System.out.println("Mammal travels");
   }
   public int noOfLegs() {
      return 0;
   }
   public static void main(String args[]) {
      MammalInt m = new MammalInt();
      m.eat();
      m.travel();
   }
}
```

This will produce the following result −

Output

```
Mammal eats
Mammal travels
```

When overriding methods defined in interfaces, there are several rules to be followed −

- Checked exceptions should not be declared on implementation methods other than the ones declared by the interface method or subclasses of those declared by the interface method.
- The signature of the interface method and the same return type or subtype should be maintained when overriding the methods.
- An implementation class itself can be abstract and if so, interface methods need not be implemented.

When implementation interfaces, there are several rules −

- A class can implement more than one interface at a time.
- A class can extend only one class, but implement many interfaces.

- An interface can extend another interface, in a similar way as a class can extend another class.

Extending Interfaces

An interface can extend another interface in the same way that a class can extend another class. The **extends** keyword is used to extend an interface, and the child interface inherits the methods of the parent interface.

The following Sports interface is extended by Hockey and Football interfaces.

Example

```
// Filename: Sports.java

public interface Sports {

   public void setHomeTeam(String name);

   public void setVisitingTeam(String name);

}


// Filename: Football.java

public interface Football extends Sports {

   public void homeTeamScored(int points);

   public void visitingTeamScored(int points);

   public void endOfQuarter(int quarter);

}

// Filename: Hockey.java

public interface Hockey extends Sports {

   public void homeGoalScored();

   public void visitingGoalScored();

   public void endOfPeriod(int period);

   public void overtimePeriod(int ot);

}
```

The Hockey interface has four methods, but it inherits two from Sports; thus, a class that implements Hockey needs to implement all six methods. Similarly, a class that implements Football needs to define the three methods from Football and the two methods from Sports.

Extending Multiple Interfaces

A Java class can only extend one parent class. Multiple inheritance is not allowed. Interfaces are not classes, however, and an interface can extend more than one parent interface.

The extends keyword is used once, and the parent interfaces are declared in a comma-separated list.

For example, if the Hockey interface extended both Sports and Event, it would be declared as −

Example

public interface Hockey extends Sports, Event

Code:

3.1 Define the following classes/ interfaces with the help of above shortcuts

1. Person(id, name, dateOfBirth, age, street, city, pin : default and parameterized constructors and setters and getters)

Code :

```
class Person {
   int dateOfBirth, age, id, pin;
   String name, street, city;
   Person() {
   }
   Person(int a, int b, int c, int d, String s,
                         String s2, String s3) {
      this.dateOfBirth = c;
      this.age = b;
      this.id = a;
      this.pin = d;
      this.name = s;
      this.street = s2;
      this.city = s3;
   }
   public int getDateOfBirth() {
      return dateOfBirth;
   }
   public void setDateOfBirth(int dateOfBirth) {
      this.dateOfBirth = dateOfBirth;
   }
   public String getCity() {
      return city;
   }
   public void setCity(String city) {
      this.city = city;
   }
   public String getStreet() {
      return street;
   }
   public void setStreet(String street) {
      this.street = street;
   }
   public int getPin() {
      return pin;
   }
   public void setPin(int pin) {
      this.pin = pin;
   }
   public String getName() {
      return name;
   }
   public void setName(String name) {
      this.name = name;
   }
   public int getId() {
      return id;
   }
   public void setId(int id) {
      this.id = id;
   }
   public int getAge() {
      return age;
   }

   void setAge(int a) {
      age = a;
   }

}
```

2. Department(id, name, dateOfEstablishment, headOfficeLocation, headId, numberOfEmployees : default and parameterized constructors and setters and getters)

```
class Department {
   int id, headId, numberOfEmployees, dateOfEstablishment;
   String headOfficeLocation;
   String name;
   Department() {
   }
   public Department(int id, int headId, int numberOfEmployees, int
dateOfEstablishment, String headOfficeLocation, String name) {
      this.id = id;
      this.headId = headId;
      this.numberOfEmployees = numberOfEmployees;
      this.dateOfEstablishment = dateOfEstablishment;
      this.headOfficeLocation = headOfficeLocation;
      this.name = name;
   }
   public int getId() {
      return id;
   }
   public void setId(int id) {
      this.id = id;
   }
   public int getHeadId() {
      return headId;
   }
   public void setHeadId(int headId) {
      this.headId = headId;
   }
   public int getNumberOfEmployees() {
      return numberOfEmployees;
   }
   public void setNumberOfEmployees(int numberOfEmployees) {
      this.numberOfEmployees = numberOfEmployees;
   }
   public int getDateOfEstablishment() {
      return dateOfEstablishment;
   }
   public void setDateOfEstablishment(int dateOfEstablishment) {
      this.dateOfEstablishment = dateOfEstablishment;
   }
   public String getHeadOfficeLocation() {
      return headOfficeLocation;
   }
   public void setHeadOfficeLocation(String headOfficeLocation) {
      this.headOfficeLocation = headOfficeLocation;
   }
```

3. Point(x, y, z : default and parameterized constructors and setters and getters)

Code :

```
class Point {
    int x, y, z;
    Point() {    }
    public Point(int x, int y, int z) {
        this.x = x;
        this.y = y;
        this.z = z;
    }
    public int getX() {
        return x;
    }
    public void setX(int x) {
        this.x = x;
    }
    public int getY() {
        return y;
    }
    public void setY(int y) {
        this.y = y;
    }

    public int getZ() {
        return z;
    }

    public void setZ(int z) {
        this.z = z;
    }

}
```

4. Vehicle(registrationNumber, rcBookNumber, manufacturer, numberOfWheels, vehicleType, model, numberOfSeats : default and parameterized constructors and setters and getters)

Code :

```
class Vehicle implements Taxable {
    int registrationNumber, rcBookNumber, manufacturer, numberOfWheels, numberOfSeats;
    String vehicleType, model, name;
    int cost;

    Vehicle() {    }

    public Vehicle(int registrationNumber, int rcBookNumber, int manufacturer, int numberOfWheels, int numberOfSeats, String vehicleType, String model) {
        this.registrationNumber = registrationNumber;
        this.rcBookNumber = rcBookNumber;
        this.manufacturer = manufacturer;
        this.numberOfWheels = numberOfWheels;
        this.numberOfSeats = numberOfSeats;
        this.vehicleType = vehicleType;
        this.model = model;
    }

    public int getCost() {
        return cost;
    }
    public void setCost(int cost) {
        this.cost = cost;
    }
    public String getName() {
        return name;
    }
    public void setName(String name) {
        this.name = name;
    }
    public int getRegistrationNumber() {
        return registrationNumber;
    }
    public void setRegistrationNumber(int registrationNumber) {
        this.registrationNumber = registrationNumber;
    }
    public int getRcBookNumber() {
        return rcBookNumber;
    }
    public void setRcBookNumber(int rcBookNumber) {
        this.rcBookNumber = rcBookNumber;
    }
    public int getManufacturer() {
        return manufacturer;
    }
    public void setManufacturer(int manufacturer) {
        this.manufacturer = manufacturer;
    }
    public int getNumberOfWheels() {
        return numberOfWheels;
    }
    public void setNumberOfWheels(int numberOfWheels) {
        this.numberOfWheels = numberOfWheels;
    }
    public int getNumberOfSeats() {
        return numberOfSeats;
    }
    public void setNumberOfSeats(int numberOfSeats) {
        this.numberOfSeats = numberOfSeats;
    }
    public String getVehicleType() {
        return vehicleType;
    }
    public void setVehicleType(String vehicleType) {
        this.vehicleType = vehicleType;
    }
    public String getModel() {
        return model;
    }
    public void setModel(String model) {
        this.model = model;
    }
    public int cost() {
        int cost = getCost();
        return cost;
    }
    public int percentGST() {
        float a = 0.18f;
        int percentGST = (int) (getCost() + (getCost() * a));
        return percentGST;
    }
}
```

5. Laptop(imeiNumber, processorName, processorSpeed, primaryMemoryType, primaryMemoryCapacity, secondaryStorageType, secondaryStorageCapaciry, screenResolution, screenType, isLED, listOfPorts, osInstalled : default and parameterized constructors and setters and getters)

Code :

```
class Laptop extends Gadget {
   int imeiNumber;
   String processorName, primaryMemoryType, secondaryStorageType, screenType;
   boolean isLED, osInstalled;
   float processorSpeed, primaryMemoryCapacity, secondaryStorageCapaciry, screenResolution;
   String listOfPorts;
   int cost;
   Laptop() {    }
   public Laptop(int imeiNumber, String processorName, String primaryMemoryType, String secondaryStorageType, String screenType, boolean isLED,
boolean osInstalled, float processorSpeed, float primaryMemoryCapacity, float secondaryStorageCapaciry, float screenResolution, String listOfPorts) {
      this.imeiNumber = imeiNumber;
      this.processorName = processorName;
      this.primaryMemoryType = primaryMemoryType;
      this.secondaryStorageType = secondaryStorageType;
      this.screenType = screenType;
      this.isLED = isLED;
      this.osInstalled = osInstalled;
      this.processorSpeed = processorSpeed;
      this.primaryMemoryCapacity = primaryMemoryCapacity;
      this.secondaryStorageCapaciry = secondaryStorageCapaciry;
      this.screenResolution = screenResolution;
      this.listOfPorts = listOfPorts;
   }
   public boolean isLED() {
      return isLED;
   }
   public void setLED(boolean LED) {
      isLED = LED;
   }
   public boolean isOsInstalled() {
      return osInstalled;
   }
   public int getCost() {
      return cost;
   }
   public void setCost(int cost) {
      this.cost = cost;
   }
   public int getImeiNumber() {
      return imeiNumber;
   }
   public void setImeiNumber(int imeiNumber) {
      this.imeiNumber = imeiNumber;
   }
   public String getProcessorName() {
      return processorName;
   }
   public void setProcessorName(String processorName) {
      this.processorName = processorName;
   }
   public String getPrimaryMemoryType() {
      return primaryMemoryType;
   }
   public void setPrimaryMemoryType(String primaryMemoryType) {
      this.primaryMemoryType = primaryMemoryType;
   }
   public String getSecondaryStorageType() {
      return secondaryStorageType;
   }
   public void setSecondaryStorageType(String secondaryStorageType) {
      this.secondaryStorageType = secondaryStorageType;
   }
   public String getScreenType() {
      return screenType;
   }
   public void setScreenType(String screenType) {
      this.screenType = screenType;
   }
   public boolean getIsLED() {
      return isLED;
   }
   public void setIsLED(Boolean isLED) {
      this.isLED = isLED;
   }
   public boolean getOsInstalled() {
      return osInstalled;
   }
   public void setOsInstalled(boolean osInstalled) {
      this.osInstalled = osInstalled;
   }
   public void setOsInstalled(Boolean osInstalled) {
      this.osInstalled = osInstalled;
   }
   public float getProcessorSpeed() {
      return processorSpeed;
   }
   public void setProcessorSpeed(float processorSpeed) {
      this.processorSpeed = processorSpeed;
   }
   public float getPrimaryMemoryCapacity() {
      return primaryMemoryCapacity;
   }
   public void setPrimaryMemoryCapacity(float primaryMemoryCapacity) {
      this.primaryMemoryCapacity = primaryMemoryCapacity;
   }
   public float getSecondaryStorageCapaciry() {
      return secondaryStorageCapaciry;
   }
   public void setSecondaryStorageCapaciry(float
secondaryStorageCapaciry) {
      this.secondaryStorageCapaciry = secondaryStorageCapaciry;
   }
   public float getScreenResolution() {
      return screenResolution;
   }
   public void setScreenResolution(float screenResolution) {
      this.screenResolution = screenResolution;
   }
   public String getListOfPorts() {
      return listOfPorts;
   }
   public void setListOfPorts(String listOfPorts) {
      this.listOfPorts = listOfPorts;
   }
```

```java
    void print() {
        System.out.println("emi no is " + getImeiNumber() + "\n Processor name is: " + getProcessorName() + "\n led :" + getIsLED() + "\n Ports are :" +
getListOfPorts() + "\n OS :" + getOsInstalled() + "\n Meomory Capacity :" + getPrimaryMemoryCapacity());
    }
    public int cost() {
        int cost = getCost();
        return cost;
    }
    public int percentGST() {
        float a = 0.18f;
        int percentGST = (int) (getCost() + (getCost() * a));
        return percentGST;
    }
}
```

6. interface Taxable(public int cost(), public intpercentGST())

Code :

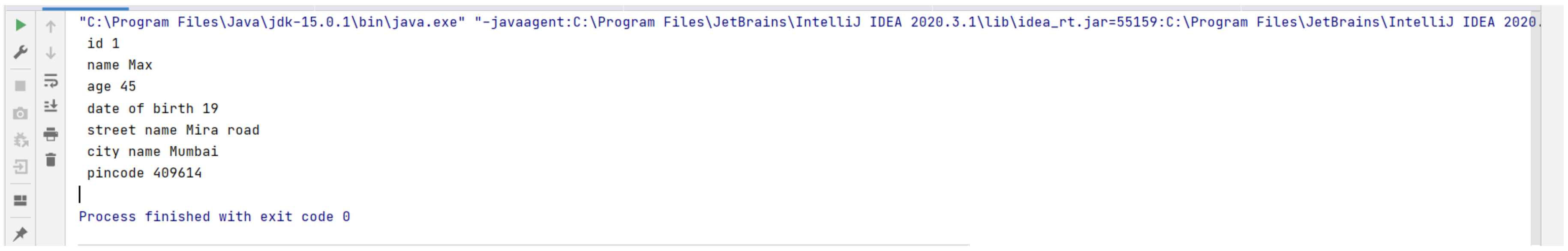
```java
interface Taxable {
    int cost();
    int percentGST();
}
```

3.2 Check whether feature of Encapsulation has been followed in 3.1. If not make necessary changes.

Code :

```java
public class Main {
    public static void main(String[] args) {
        Person person = new Person();
        person.setAge(45);
        person.setId(01);
        person.setName("Max");
        person.setDateOfBirth(19);
        person.setStreet("Mira road");
        person.setCity("Mumbai");
        person.setPin(409614);
        System.out.println(" id " + person.getId() + "\n name " + person.getName() + "\n age " + person.getAge() + "\n date of birth " + person.getDateOfBirth()
+ "\n street name " + person.getStreet() + "\n city name " + person.getCity() + "\n pincode " + person.getPin());
    }
}
```

Output :



```
"C:\Program Files\Java\jdk-15.0.1\bin\java.exe" "-javaagent:C:\Program Files\JetBrains\IntelliJ IDEA 2020.3.1\lib\idea_rt.jar=55159:C:\Program Files\JetBrains\IntelliJ IDEA 2020.
 id 1
 name Max
 age 45
 date of birth 19
 street name Mira road
 city name Mumbai
 pincode 409614

Process finished with exit code 0
```

3.3 Define classes Car, Train and Truck with necessary member fields, constructors and methods. Make them extend class Vehicle.

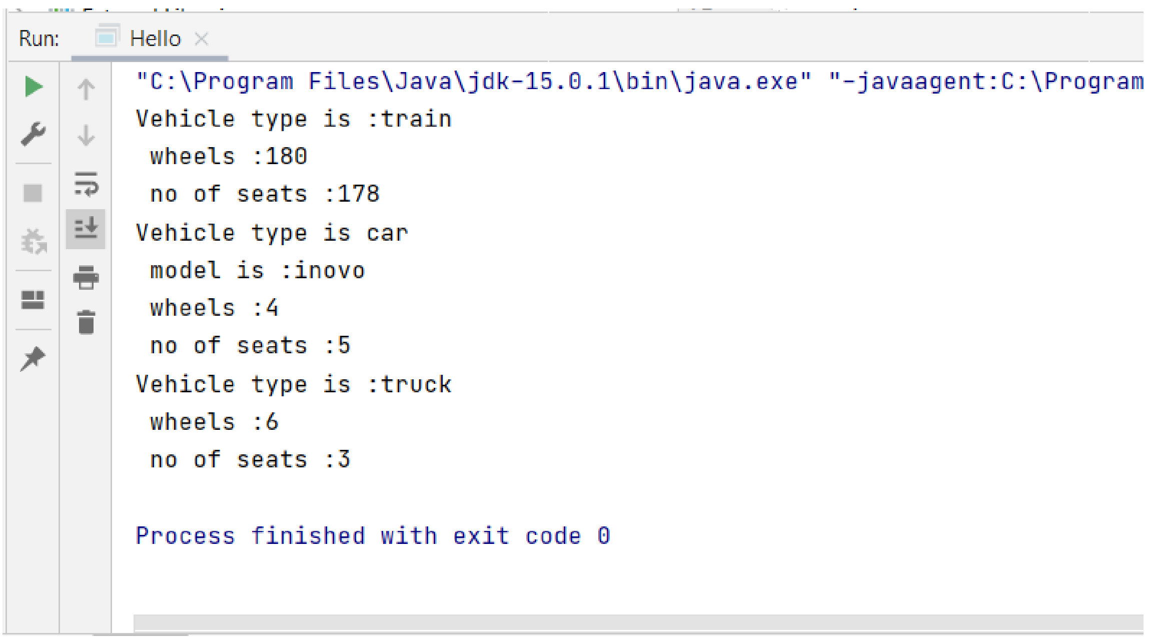
Code :

```java
class Car extends Vehicle {
    int cost;
    @Override
    public int getCost() {
        return cost;
    }
    @Override
    public void setCost(int cost) {
        this.cost = cost;
    }
    void show() {
        System.out.println("vehicle type is " + getVehicleType() + "\n model is :" + getModel() + "\n wheels :" + getNumberOfWheels()
                + "\n no of seats :" + getNumberOfSeats() + "");
    }
    void disp() {
        System.out.println("name is :" + getName());
    }
    public int cost() {
        int cost = getCost();
        return cost;
    }
    public int percentGST() {
        float a = 0.18f;
        int percentGST = (int) (getCost() + (getCost() * a));
        return percentGST;
    }
}

class Train extends Vehicle {
    void show() {
        System.out.println("vehicle type is :" + getVehicleType() + "\n wheels :" + getNumberOfWheels()
                + "\n no of seats :" + getNumberOfSeats() + "");
    }
}

class Truck extends Vehicle {
    void show() {
        System.out.println("vehicle type is :" + getVehicleType() + "\n wheels :" + getNumberOfWheels()
                + "\n no of seats :" + getNumberOfSeats() + "");
    }
}
public static void main(String[] args) {
    Train h = new Train();
    h.setVehicleType("train");
    h.setNumberOfSeats(178);
    h.setNumberOfWheels(180);
    h.setRegistrationNumber(90764);
    h.show();
    Car c = new Car();
    c.setVehicleType("car");
    c.setModel("inovo");
    c.setNumberOfSeats(5);
    c.setNumberOfWheels(4);
    c.setRegistrationNumber(90671);
    c.show();
    Truck t = new Truck();
    t.setVehicleType("truck");
    t.setNumberOfSeats(3);
    t.setNumberOfWheels(6);
    t.show();
}
```

Output:



```
"C:\Program Files\Java\jdk-15.0.1\bin\java.exe" "-javaagent:C:\Program
Vehicle type is :train
 wheels :180
 no of seats :178
Vehicle type is car
 model is :inovo
 wheels :4
 no of seats :5
Vehicle type is :truck
 wheels :6
 no of seats :3

Process finished with exit code 0
```

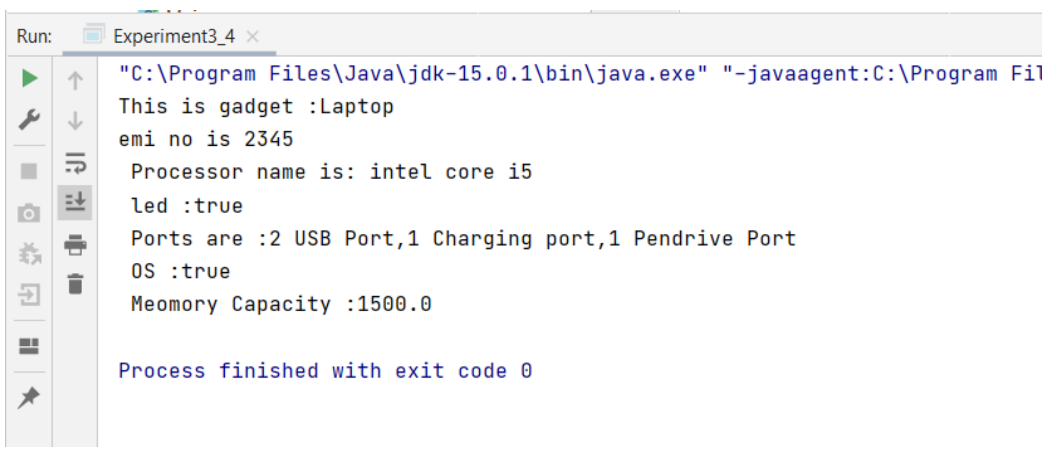
3.4 Define a class Gadget with necessary member fields, constrictors and methods. Modify the class Laptop to extend the class Gadget.

Code :

```
public class Gadget implements Taxable {
    static int gadgetcount = 0;
    String gadgetName;
    int cost;
    {
        gadgetcount += 1;
    }
    Gadget() {
    }
    public Gadget(String gadgetName) {
        this.gadgetName = gadgetName;
    }

    void disp() {
        System.out.println("The object of a class Gadget is initialized " + gadgetcount + " times");
    }
    public String getGadgetName() {
        return gadgetName;
    }
    public void setGadgetName(String gadgetName) {
        this.gadgetName = gadgetName;
    }
    public int getCost() {
        return cost;
    }
    public void setCost(int cost) {
        this.cost = cost;
    }
    void Show() {
        System.out.println("This is gadget :" + getGadgetName());
    }
    public int cost() {
        int cost = getCost();
        return cost;
    }
    public int percentGST() {
        float a = 0.18f;
        int percentGST = (int) (getCost() + (getCost() * a));
        return percentGST;
    }
}
```

Output:



```
"C:\Program Files\Java\jdk-15.0.1\bin\java.exe" "-javaagent:C:\Program Fil
This is gadget :Laptop
emi no is 2345
 Processor name is: intel core i5
 led :true
 Ports are :2 USB Port,1 Charging port,1 Pendrive Port
 OS :true
 Meomory Capacity :1500.0

Process finished with exit code 0
```

Main method :

```
public static void main(String args[]) {
    Laptop l = new Laptop();
    l.setGadgetName("Laptop");
    l.setImeiNumber(2345);
    l.setIsLED(true);
    l.setListOfPorts("2 USB Port,1 Charging port,1 Pendrive Port");
    l.setOsInstalled(true);
    l.setPrimaryMemoryCapacity(1500);
    l.setProcessorName("intel core i5");
    l.Show();
    l.print();
}
```

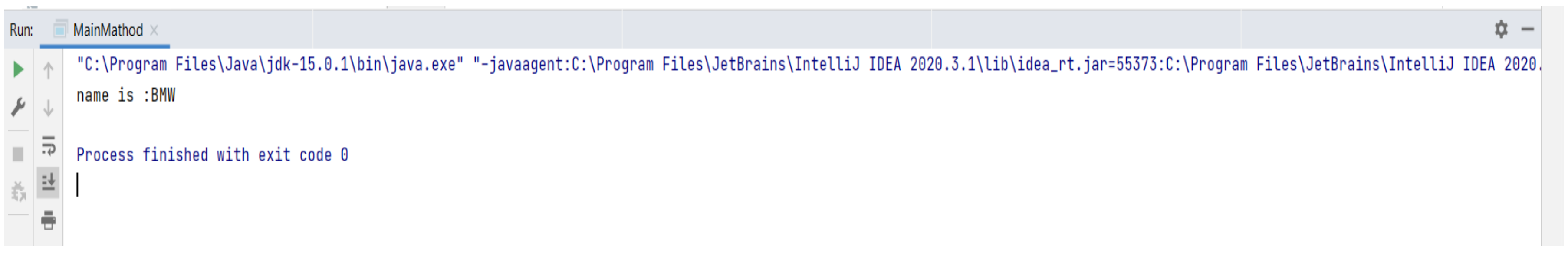
3.5 In main method, declare a reference variable vehicle of class Vehicle and create an object of class Car which will be referenced by vehicle. Call getName() method on the object. (Hint: Reference Variable Casting)

Code :

```
class Vehicle implements Taxable {
    int registrationNumber, rcBookNumber, manufacturer, numberOfWheels, numberOfSeats;
    String vehicleType, model, name;
    int cost;
    Vehicle() {
    }
    public Vehicle(int registrationNumber, int rcBookNumber, int manufacturer, int numberOfWheels, int numberOfSeats, String vehicleType, String model) {
        this.registrationNumber = registrationNumber;
        this.rcBookNumber = rcBookNumber;
        this.manufacturer = manufacturer;
        this.numberOfWheels = numberOfWheels;
        this.numberOfSeats = numberOfSeats;
        this.vehicleType = vehicleType;
        this.model = model;
    }
    public int getCost() {
        return cost;
    }
    public void setCost(int cost) {
        this.cost = cost;
    }
    public String getName() {
        return name;
    }
    public void setName(String name) {
        this.name = name;
    }
    public int getRegistrationNumber() {
        return registrationNumber;
    }
    public void setRegistrationNumber(int registrationNumber) {
        this.registrationNumber = registrationNumber;
    }
    public int getRcBookNumber() {
        return rcBookNumber;
    }
    public void setRcBookNumber(int rcBookNumber) {
        this.rcBookNumber = rcBookNumber;
    }
    public int getManufacturer() {
        return manufacturer;
    }
    public void setManufacturer(int manufacturer) {
        this.manufacturer = manufacturer;
    }
```

```java
    public int getNumberOfWheels() {
        return numberOfWheels;
    }
    public void setNumberOfWheels(int numberOfWheels) {
        this.numberOfWheels = numberOfWheels;
    }
    public int getNumberOfSeats() {
        return numberOfSeats;
    }
    public void setNumberOfSeats(int numberOfSeats) {
        this.numberOfSeats = numberOfSeats;
    }
    public String getVehicleType() {
        return vehicleType;
    }
    public void setVehicleType(String vehicleType) {
        this.vehicleType = vehicleType;
    }
    public String getModel() {
        return model;
    }
    public void setModel(String model) {
        this.model = model;
    }
    public int cost() {
        int cost = getCost();
        return cost;
    }
    public int percentGST() {
        float a = 0.18f;
        int percentGST = (int) (getCost() + (getCost() * a));
        return percentGST;
    }
}
class Car extends Vehicle {
    int cost;
    @Override
    public int getCost() {
        return cost;
    }
    @Override
    public void setCost(int cost) {
        this.cost = cost;
    }
    void show() {
        System.out.println("Vehicle type is " + getVehicleType() + "\n model is :" + getModel() + "\n wheels :" + getNumberOfWheels()
                + "\n no of seats :" + getNumberOfSeats() + "");
    }
    void disp() {
        System.out.println("name is :" + getName());
    }
    public int cost() {
        int cost = getCost();
        return cost;
    }
    public int percentGST() {
        float a = 0.18f;
        int percentGST = (int) (getCost() + (getCost() * a));
        return percentGST;
    }
}
public static void main(String [] args){
    Vehicle vehicle;
    Car c1=new Car();
    c1.setName("BMW");
    c1.disp();
}
```

Output :



Run: MainMathod

```
"C:\Program Files\Java\jdk-15.0.1\bin\java.exe" "-javaagent:C:\Program Files\JetBrains\IntelliJ IDEA 2020.3.1\lib\idea_rt.jar=55373:C:\Program Files\JetBrains\IntelliJ IDEA 2020.
name is :BMW

Process finished with exit code 0
```

3.6 Modify the classes Vehicle and Gadget implement the interface Taxable. Hence override respective methods.

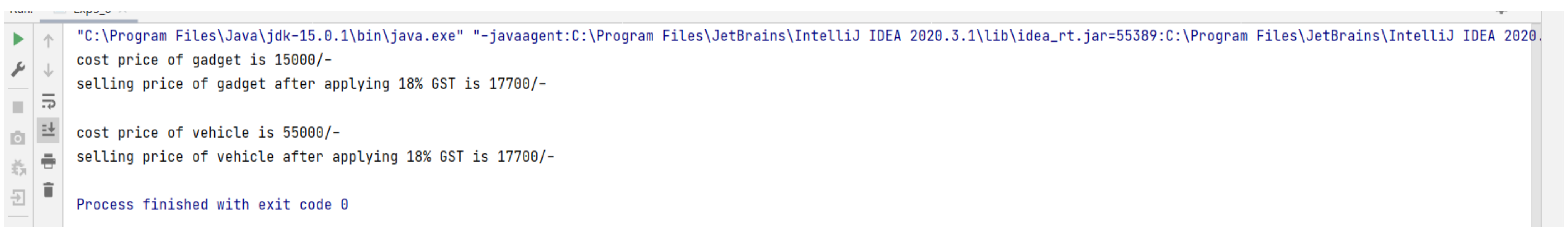
Code :

```
interface Taxable {
    int cost();
    int percentGST();
}
public class Gadget implements Taxable {
    static int gadgetcount = 0;
    String gadgetName;
    int cost;
    {
        gadgetcount += 1;
    }
    Gadget() {
    }
    public Gadget(String gadgetName) {
        this.gadgetName = gadgetName;
    }
    void disp() {
        System.out.println("The object of a class Gadget is initialized " + gadgetcount + " times");
    }
    public String getGadgetName() {
        return gadgetName;
    }
    public void setGadgetName(String gadgetName) {
        this.gadgetName = gadgetName;
    }
    public int getCost() {
        return cost;
    }
    public void setCost(int cost) {
        this.cost = cost;
    }
    void Show() {
        System.out.println("This is gadget :" + getGadgetName())
    }
    public int cost() {
        int cost = getCost();
        return cost;
    }
    public int percentGST() {
        float a = 0.18f;
        int percentGST = (int) (getCost() + (getCost() * a));
        return percentGST;
    }
}
class Vehicle implements Taxable {
    int registrationNumber, rcBookNumber, manufacturer, numberOfWheels, numberOfSeats;
    String vehicleType, model, name;
    int cost;
    Vehicle() {
    }
    public Vehicle(int registrationNumber, int rcBookNumber, int manufacturer, int numberOfWheels, int numberOfSeats, String vehicleType, String model) {
        this.registrationNumber = registrationNumber;
        this.rcBookNumber = rcBookNumber;
        this.manufacturer = manufacturer;
        this.numberOfWheels = numberOfWheels;
        this.numberOfSeats = numberOfSeats;
        this.vehicleType = vehicleType;
        this.model = model;
    }
    public int getCost() {
        return cost;
    }
    public void setCost(int cost) {
        this.cost = cost;
    }
    public String getName() {
        return name;
    }
    public void setName(String name) {
        this.name = name;
    }
```

```java
    public int getRegistrationNumber() {
        return registrationNumber;
    }
    public void setRegistrationNumber(int registrationNumber) {
        this.registrationNumber = registrationNumber;
    }
    public int getRcBookNumber() {
        return rcBookNumber;
    }
    public void setRcBookNumber(int rcBookNumber) {
        this.rcBookNumber = rcBookNumber;
    }
    public int getManufacturer() {
        return manufacturer;
    }
    public void setManufacturer(int manufacturer) {
        this.manufacturer = manufacturer;
    }
    public int getNumberOfWheels() {
        return numberOfWheels;
    }
    public void setNumberOfWheels(int numberOfWheels) {
        this.numberOfWheels = numberOfWheels;
    }
    public int getNumberOfSeats() {
        return numberOfSeats;
    }
    public void setNumberOfSeats(int numberOfSeats) {
        this.numberOfSeats = numberOfSeats;
    }

    public String getVehicleType() {
        return vehicleType;
    }
    public void setVehicleType(String vehicleType) {
        this.vehicleType = vehicleType;
    }
    public String getModel() {
        return model;
    }
    public void setModel(String model) {
        this.model = model;
    }
    public int cost() {
        int cost = getCost();
        return cost;
    }
    public int percentGST() {
        float a = 0.18f;
        int percentGST = (int) (getCost() + (getCost() * a));
        return percentGST;
    }
}
public static void main(String args[]) {
    Gadget g = new Gadget();
    g.setCost(15000);
    Vehicle v = new Vehicle();
    v.setCost(55000);
    System.out.println("cost price of gadget is " + g.cost() + "/-");
    System.out.println("selling price of gadget after applying 18% GST is " + g.percentGST() + "/-");
    System.out.println();
    System.out.println("cost price of vehicle is " + v.cost() + "/-");
    System.out.println("selling price of vehicle after applying 18% GST is " + g.percentGST() + "/-");
}
```

Output :



```
"C:\Program Files\Java\jdk-15.0.1\bin\java.exe" "-javaagent:C:\Program Files\JetBrains\IntelliJ IDEA 2020.3.1\lib\idea_rt.jar=55389:C:\Program Files\JetBrains\IntelliJ IDEA 2020.
cost price of gadget is 15000/-
selling price of gadget after applying 18% GST is 17700/-

cost price of vehicle is 55000/-
selling price of vehicle after applying 18% GST is 17700/-

Process finished with exit code 0
```

3.7 Modify the classes Car and Laptop to override the implemented methods in 3.6.

Code :

```
class Laptop extends Gadget {
   int imeiNumber;
   String processorName, primaryMemoryType, secondaryStorageType, screenType;
   boolean isLED, osInstalled;
   float processorSpeed, primaryMemoryCapacity, secondaryStorageCapaciry, screenResolution;
   String listOfPorts;
   int cost;
   Laptop() {
   }
   public Laptop(int imeiNumber, String processorName, String primaryMemoryType, String secondaryStorageType, String screenType, boolean isLED,
boolean osInstalled, float processorSpeed, float primaryMemoryCapacity, float secondaryStorageCapaciry, float screenResolution, String listOfPorts) {
      this.imeiNumber = imeiNumber;
      this.processorName = processorName;
      this.primaryMemoryType = primaryMemoryType;
      this.secondaryStorageType = secondaryStorageType;
      this.screenType = screenType;
      this.isLED = isLED;
      this.osInstalled = osInstalled;
      this.processorSpeed = processorSpeed;
      this.primaryMemoryCapacity = primaryMemoryCapacity;
      this.secondaryStorageCapaciry = secondaryStorageCapaciry;
      this.screenResolution = screenResolution;
      this.listOfPorts = listOfPorts;
   }
   public boolean isLED() {
      return isLED;
   }
   public void setLED(boolean LED) {
      isLED = LED;
   }
   public boolean isOsInstalled() {
      return osInstalled;
   }
   public int getCost() {
      return cost;
   }
   public void setCost(int cost) {
      this.cost = cost;
   }
   public int getImeiNumber() {
      return imeiNumber;
   }
   public void setImeiNumber(int imeiNumber) {
      this.imeiNumber = imeiNumber;
   }
   public String getProcessorName() {
      return processorName;
   }
   public void setProcessorName(String processorName) {
      this.processorName = processorName;
   }
   public String getPrimaryMemoryType() {
      return primaryMemoryType;
   }
   public void setPrimaryMemoryType(String primaryMemoryType) {
      this.primaryMemoryType = primaryMemoryType;
   }
   public String getSecondaryStorageType() {
      return secondaryStorageType;
   }
    public void setSecondaryStorageType(String secondaryStorageType) {
      this.secondaryStorageType = secondaryStorageType;
   }
   public String getScreenType() {
      return screenType;x
  }
   public void setScreenType(String screenType) {
      this.screenType = screenType;
   }
   public boolean getIsLED() {
      return isLED;
   }
public void setIsLED(Boolean isLED) {
      this.isLED = isLED;
   }
   public boolean getOsInstalled() {
      return osInstalled;
   }
   public void setOsInstalled(boolean osInstalled) {
      this.osInstalled = osInstalled;
   }
   public void setOsInstalled(Boolean osInstalled) {
      this.osInstalled = osInstalled;
   }
   public float getProcessorSpeed() {
      return processorSpeed;
   }
   public void setProcessorSpeed(float processorSpeed) {
      this.processorSpeed = processorSpeed;
   }
   public float getPrimaryMemoryCapacity() {
      return primaryMemoryCapacity;
   }
   public void setPrimaryMemoryCapacity(float primaryMemoryCapacity) {
      this.primaryMemoryCapacity = primaryMemoryCapacity;
   }
   public float getSecondaryStorageCapaciry() {
      return secondaryStorageCapaciry;
   }
   public void setSecondaryStorageCapaciry(float secondaryStorageCapaciry) {
      this.secondaryStorageCapaciry = secondaryStorageCapaciry;
   }
   public float getScreenResolution() {
      return screenResolution;
   }
   public void setScreenResolution(float screenResolution) {
      this.screenResolution = screenResolution;
   }
   public String getListOfPorts() {
      return listOfPorts;
   }
   public void setListOfPorts(String listOfPorts) {
      this.listOfPorts = listOfPorts;
   }
```

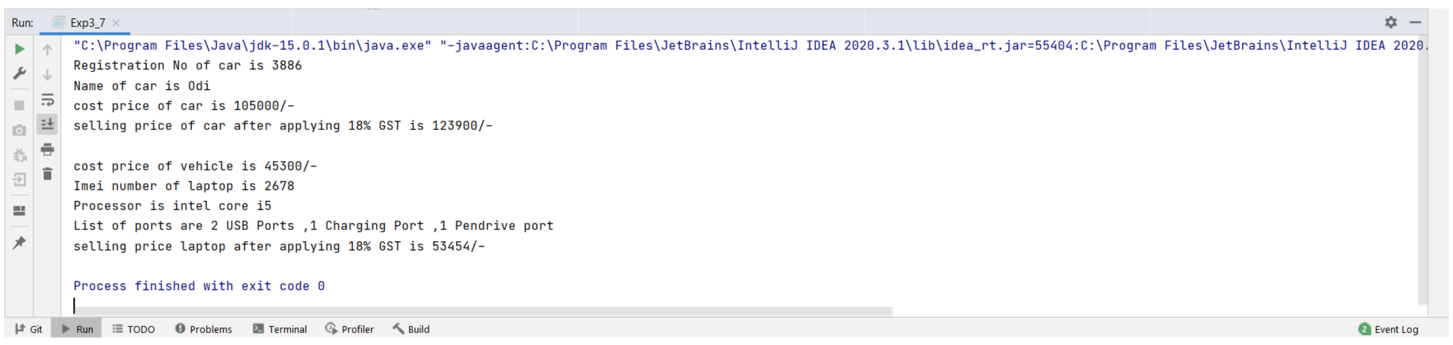
```
    void print() {
    System.out.println("emi no is " + getImeiNumber() + "\n Processor name is: " + getProcessorName() + "\n led :" + getIsLED() + "\n Ports are :" +
getListOfPorts() + "\n OS :" + getOsInstalled() + "\n Meomory Capacity :" + getPrimaryMemoryCapacity());
  }
  public int cost() {
    int cost = getCost();
    return cost;
  }
  public int percentGST() {
    float a = 0.18f;
    int percentGST = (int) (getCost() + (getCost() * a));
    return percentGST;
  }
}


class Car extends Vehicle {
  int cost;
  @Override
  public int getCost() {
    return cost;
  }
  @Override
  public void setCost(int cost) {
    this.cost = cost;
  }
  void show() {
    System.out.println("Vehicle type is " + getVehicleType() + "\n model is :" + getModel() + "\n wheels :" + getNumberOfWheels()
        + "\n no of seats :" + getNumberOfSeats() + "");
  }
  void disp() {
    System.out.println("name is :" + getName());
  }
  public int cost() {
    int cost = getCost();
    return cost;
  }
  public int percentGST() {
    float a = 0.18f;
    int percentGST = (int) (getCost() + (getCost() * a));
    return percentGST;
  }
}
```

Main method :

```
public static void main(String args[]){
  Car c=new Car();
  c.setName("Odi");
  c.setCost(105000);
  c.setRegistrationNumber(07456);
  Laptop l=new Laptop();
  l.setCost(45300);
  l.setProcessorName("intel core i5");
  l.setImeiNumber(2678);
  l.setListOfPorts("2 USB Ports ,1 Charging Port ,1 Pendrive port ");
  System.out.println("Registration No of car is "+c.getRegistrationNumber()+"\nName of car is "+ c.getName()+"\ncost price of car is "+c.cost()+"/-");
  System.out.println("selling price of car after applying 18% GST is "+c.percentGST()+"/-");
  System.out.println();
  System.out.println("cost price of vehicle is "+l.cost()+"/-");
  System.out.println("Imei number of laptop is "+l.getImeiNumber()+"\nProcessor is "+l.getProcessorName()+"\nList of ports are
"+l.getListOfPorts()+"\nselling price laptop after applying 18% GST is "+l.percentGST()+"/-");
}
```

Output :



```
"C:\Program Files\Java\jdk-15.0.1\bin\java.exe" "-javaagent:C:\Program Files\JetBrains\IntelliJ IDEA 2020.3.1\lib\idea_rt.jar=55404:C:\Program Files\JetBrains\IntelliJ IDEA 2020.
Registration No of car is 3886
Name of car is Odi
cost price of car is 105000/-
selling price of car after applying 18% GST is 123900/-

cost price of vehicle is 45300/-
Imei number of laptop is 2678
Processor is intel core i5
List of ports are 2 USB Ports ,1 Charging Port ,1 Pendrive port
selling price laptop after applying 18% GST is 53454/-

Process finished with exit code 0
```

3.8 Modify the class Gadget to add a data member gadgetCount such that its value will incremented as soon as a new object is initialized. Create 5 objects of the class Print its value after initializing each object.

Code :

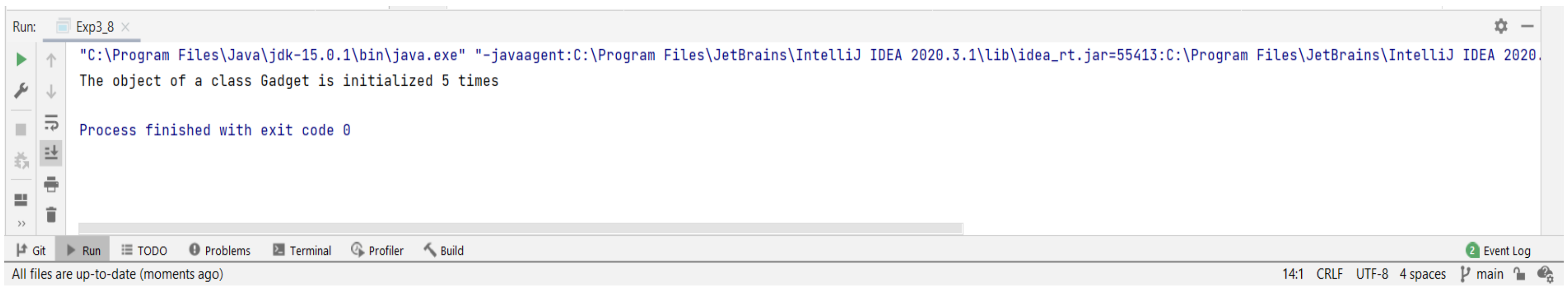
```
public class Gadget implements Taxable {
    static int gadgetcount = 0;
    String gadgetName;
    int cost;
    {
        gadgetcount += 1;
    }
    Gadget() {
    }
    public Gadget(String gadgetName) {
        this.gadgetName = gadgetName;
    }
    void disp() {
        System.out.println("The object of a class Gadget is initialized " + gadgetcount + " times");
    }
    public String getGadgetName() {
        return gadgetName;
    }
    public void setGadgetName(String gadgetName) {
        this.gadgetName = gadgetName;
    }
    public int getCost() {
        return cost;
    }
    public void setCost(int cost) {
        this.cost = cost;
    }
    void Show() {
        System.out.println("This is gadget :" + getGadgetName());

    }
    public int cost() {
        int cost = getCost();
        return cost;
    }
    public int percentGST() {
        float a = 0.18f;
        int percentGST = (int) (getCost() + (getCost() * a));
        return percentGST;
    }
}
```

Main method :

```
  public static void main(String args[])
    {
        Gadget g=new Gadget();
        Gadget g1=new Gadget();
        Gadget g2=new Gadget();
        Gadget g3=new Gadget();
        Gadget g4=new Gadget();
        g.disp();
    }
}
```

Output :



```
"C:\Program Files\Java\jdk-15.0.1\bin\java.exe" "-javaagent:C:\Program Files\JetBrains\IntelliJ IDEA 2020.3.1\lib\idea_rt.jar=55413:C:\Program Files\JetBrains\IntelliJ IDEA 2020.
The object of a class Gadget is initialized 5 times

Process finished with exit code 0
```

Run:
Exp3_8
Git
Run
TODO
Problems
Terminal
Profiler
Build
Event Log
All files are up-to-date (moments ago)
14:1 CRLF UTF-8 4 spaces main

Conclusion: Thus, we understood and executed various programs using classes, interfaces, etc. and explored various concepts related to these topics.