```java
// Object Create :-> 1

 interface Vehicle {
    public void engine();
    public void breaks();
    public void wheels();
}

// Object Create :-> 2

class All_Vehicle {
    public static void sendVehicle(Vehicle vehicle) {

        vehicle.engine();
        vehicle. breaks();
        vehicle. wheels();
    }
}

// Object Create :-> 3

 class Bus implements  Vehicle {

        static {
           System.out.println("Bus class is loaded");
           System.out.println();
         }

         Bus() {
            System.out.println("Bus class object is created");
             System.out.println();
         }

          public void engine () {
              System.out.println("Bus engine capacity is 40 kmph");
               System.out.println();
         }

           public void breaks () {
              System.out.println("Bus has two breaks");
              System.out.println();
         }

          public void wheels() {
          System.out.println("Bus will run on 4 wheels");
          System.out.println();
```

```
47      }
48  }
49
50  // Object Create :-> 4
51
52
53   class RedBus implements Vehicle {
54
55      static {
56            System.out.println("RedBus class is loaded");
57            System.out.println();
58          }
59
60          RedBus() {
61              System.out.println("RedBus class object is created");
62               System.out.println();
63          }
64
65       public void engine() {
66                System.out.println("RedBus engine capacity is 60 kmph");
67                System.out.println();
68          }
69
70
71            public void breaks () {
72                System.out.println("RedBus has also two breaks");
73                System.out.println();
74          }
75
76          public void wheels() {
77          System.out.println("RedBus will run on 8 wheels");
78          System.out.println();
79      }
80  }
81
82  // Object Create :-> 5
83
84   class Volvo implements Vehicle {
85
86      static {
87         System.out.println("Volvo class is Loaded");
88         System.out.println();
89       }
90
91           Volvo () {
92              System.out.println("Volvo object is created");
```

```
93              System.out.println();
94          }
95
96            public void engine () {
97               System.out.println("Volvo Engine capacity is 110 kmph");
98                System.out.println();
99          }
100
101           public void breaks () {
102              System.out.println("Volvo has two breaks");
103               System.out.println();
104         }
105
106         public void wheels() {
107         System.out.println("Volvo will run on also 4 wheels");
108         System.out.println();
109     }
110 }
111
112 // Object Create :-> 6
113
114  class Bike implements Vehicle {
115
116     static {
117        System.out.println("Bike class is Loaded");
118        System.out.println();
119      }
120
121         Bike () {
122            System.out.println("Bike object is created");
123            System.out.println();
124         }
125
126           public void engine () {
127              System.out.println("Bike Engine capacity is 120 kmph");
128              System.out.println();
129         }
130
131           public void breaks () {
132              System.out.println("Bike has two breaks");
133              System.out.println();
134         }
135
136         public void wheels() {
137         System.out.println("Bike will run on 2 wheels");
138         System.out.println();
```

```
139     }
140 }
141
142
143
144 // Main method:->
145
146 import java.util.Scanner;
147
148 class VehicleDriver {
149    public static void main(String[] args) throws ClassNotFoundException,
150                                                   InstantiationException,
151                                                    IllegalAccessException {
152       All_Vehicle all_vehicle = new All_Vehicle();
153
154       // Scanner API
155          Scanner scn = new Scanner(System.in);
156
157          System.out.println("Enter the Vehicle: ");
158          String vehicleName = scn.nextLine();
159
160          // Reflection API for loading & Instantiating
161
162          Class cls = Class.forName(vehicleName);
163          Object obj = cls.newInstance();
164
165          if(obj instanceof Vehicle) {
166             Vehicle vehicle = (Vehicle)obj;
167             All_Vehicle.sendVehicle(vehicle);
168    }
169
170       else {
171          System.out.println("Only pass Vehicle ");
172       }
173    }
174 }
175
176 // by shamshad
```