LAB # 9

OBJECTIVE:

To study Method Overriding and polymorphism.

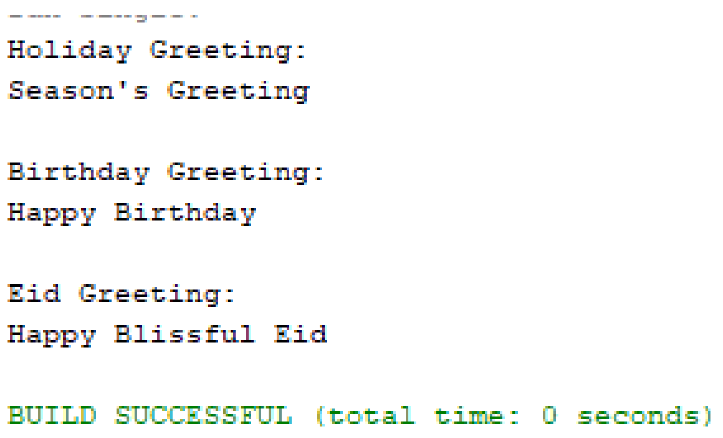
Question1:

Create a class Card, and its children classes are Valentine, Holiday, and Birthday. A card class will have a greeting () method that writes out a greeting. Each type of card contains an appropriate greeting. The Holiday card says "Season's Greetings." The Birthday card says "Happy Birthday." The Eid card says "Happy blissful Eid". Create objects of the three child classes: Holiday, Birthday and Eid and show the polymorphic behavior (call the greeting methods from each object).

Source Code:

```
class Card {
    public void greeting(){
        System.out.println("Greetings");
    }
}


 class Birthday extends Card {
     public void greeting(){
        System.out.println("Happy Birthday");
    }
}


class Eid extends Card{
    public void greeting(){
        System.out.println("Happy Blissful Eid");
    }
}

 class Holiday extends Card {
    public void greeting(){
        System.out.println("Season's Greeting");
    }
}

 class Sum  {

    public static void main(String[] args) {
        Holiday h = new Holiday();
        Birthday b = new Birthday();
        Eid e = new Eid();

        System.out.println("Holiday Greeting: ");
        h.greeting();
        System.out.println();
        System.out.println("Birthday Greeting: ");
        b.greeting();
        System.out.println();
        System.out.println("Eid Greeting: ");
        e.greeting();
        System.out.println();
    }
}
```

Output:



```
Holiday Greeting:
Season's Greeting

Birthday Greeting:
Happy Birthday

Eid Greeting:
Happy Blissful Eid

BUILD SUCCESSFUL (total time: 0 seconds)
```

Question2:

Create a super class called Car. The Car class has the following fields and methods. int speed; double regularPrice; String color; double getSalePrice(); Create a sub class of Car class and name it as Truck. The Truck class has the following fields andmethods.int weight doublegetSalePrice();//Ifweight>2000,10%discount.Otherwise,20%discount. Create a subclass of Car class and name it as Ford. The Ford class has the following fields and methods. int year; int manufacturerDiscount; double getSalePrice();//From the sale price computed from Car class, subtract the manufacturer Discount. Create MyOwnAutoShop class which contains the main() method. Perform the following within the main() method. Create two instances of the Ford and Truck class and initialize all the fields with appropriate values. Use super(...) method in the constructor for initializing the fields of the super class. Create an instance of Car class and initialize all the fields with appropriate values. Display the sale prices of all instance. Show polymorphic behavior.

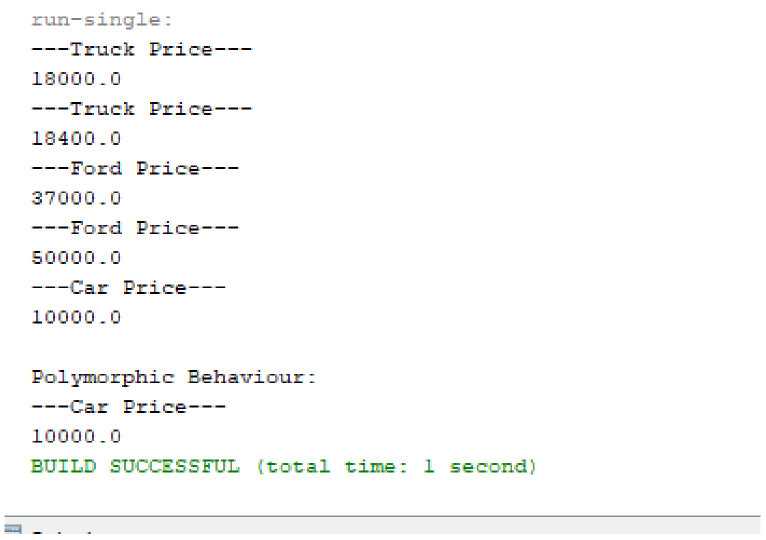
Source Code:

```
class Car {
   int speed;
   double RegularPrice;
   String Color;

   Car(int s, double rp, String c){
      speed = s;
```

```
        RegularPrice = rp;
        Color = c;
    }

    double getSalePrice(){
        System.out.println("---Car Price---");
        return RegularPrice;
    }
}

 class Truck extends Car {
    int Weight;
    Truck(int s, double rp, String c, int w) {
        super(s, rp, c);
        Weight = w;
    }

    double getSalePrice(){
        int DiscountPercentage = (this.Weight>2000)? 10: 20;
        double price = RegularPrice - (RegularPrice*DiscountPercentage)/100.0;
        System.out.println("---Truck Price---");
        return price;
    }
}

 class Ford extends Car{
    int year;
    int ManufactureDiscount;

    Ford(int s, double rp, String c, int y, int md){
        super(s, rp, c);
        year = y;
        ManufactureDiscount = md;
    }

    double getSalePrice(){
        double price = RegularPrice - ManufactureDiscount;
        System.out.println("---Ford Price---");
        return price;
    }
}
```

```
class MyOwnAutoShop {

  public static void main(String[] args) {
      Truck t1 = new Truck(200, 20000, "Black", 2100);
      Truck t2 = new Truck(220, 23000, "Red", 1800);

      Ford f1 = new Ford(800, 40000, "Black", 2010, 3000);
      Ford f2 = new Ford(1000, 55000, "Red", 2015, 5000);

      Car c = new Car(400, 10000, "White");
      System.out.println(t1.getSalePrice());
      System.out.println(t2.getSalePrice());
      System.out.println(f1.getSalePrice());
      System.out.println(f2.getSalePrice());
      System.out.println(c.getSalePrice() + "\n");
      System.out.println("Polymorphic Behaviour: ");
      Car[] arr = {t1,t2,f1,f2,c};
      int index = (int)(Math.random()*arr.length);
      System.out.println(arr[index].getSalePrice());
  }
}
```

Output:



```
run-single:
---Truck Price---
18000.0
---Truck Price---
18400.0
---Ford Price---
37000.0
---Ford Price---
50000.0
---Car Price---
10000.0

Polymorphic Behaviour:
---Car Price---
10000.0
BUILD SUCCESSFUL (total time: 1 second)
```

ASSIGNMENT

QUES 1:
Is constructor overloading polymorphism? If Yes or No, Justify your Answer.

ANSWER

No, because polymorphism works in overload, not overload. With polymorphism methods, we can collect multiple classes with the same behavior or methods and each category has its own behavior in the same way (with the same signature and name). So, if we have to come up with an alternative, it can be done in the same class that is not part of the polymorphism so, which is why overloading the builder is not a polymorphism.

QUES 2:
Figure out the type of polymorphism is method overloading and method overriding, respectively.

ANSWER

Overloading is one of the modes of application in polymorphism, especially when operators are involved. Typically, refer to a polymorphism where two or more categories are involved. While overload can also be done within the same category, we can overload the path name with several signatures (separate list of parameters). While writing above is only meant to include two or more classes.