**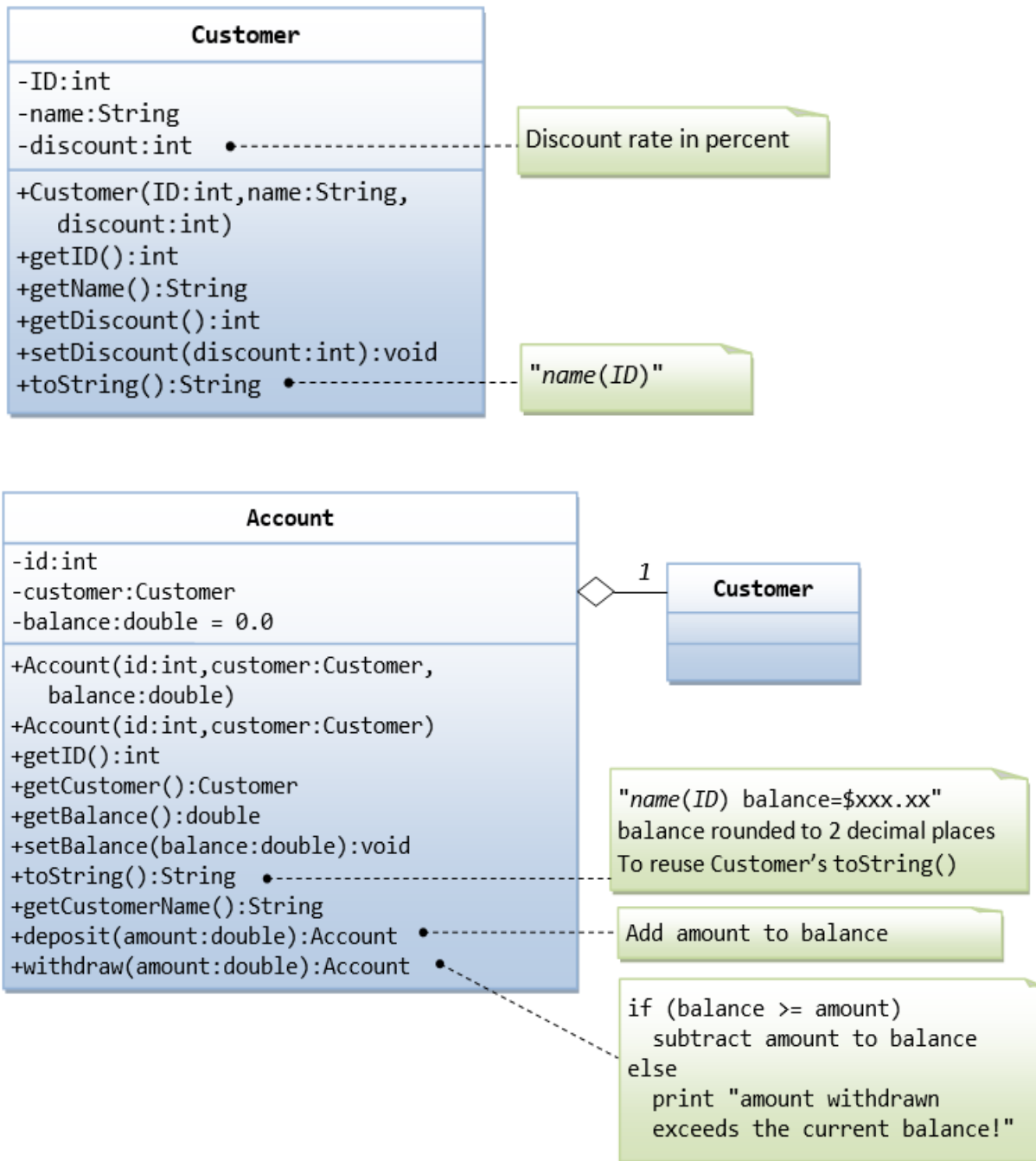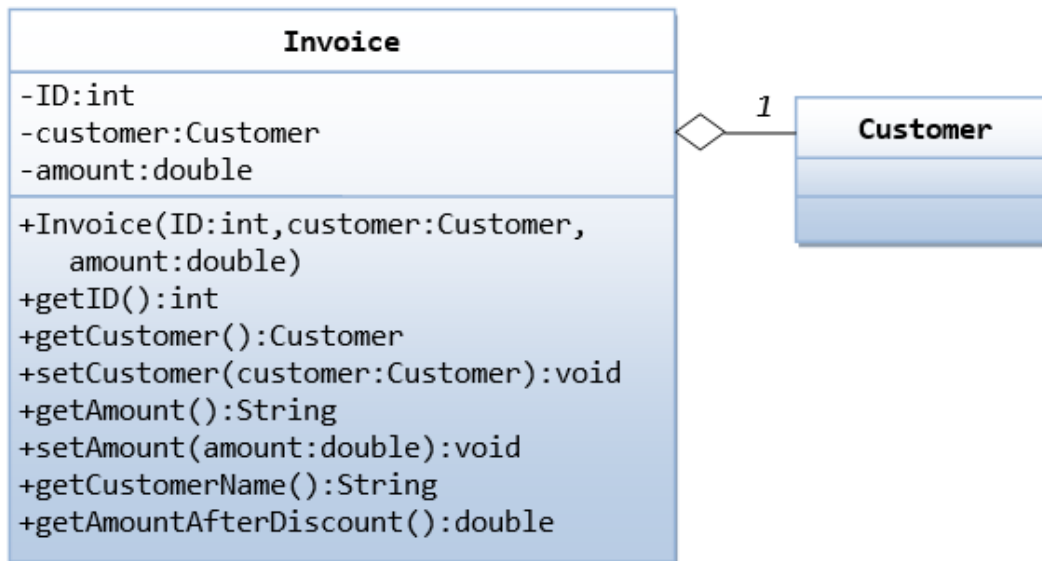Q1. Write a Java program to create Customer, Account and Invoice classes. Test the created classes by creating an object for each class.**

```
            Customer
------------------------------------
-ID:int
-name:String
-discount:int        •----------------   Discount rate in percent
------------------------------------
+Customer(ID:int,name:String,
   discount:int)
+getID():int
+getName():String
+getDiscount():int
+setDiscount(discount:int):void
+toString():String   •----------------   "name(ID)"
```

```
              Account
------------------------------------
-id:int
-customer:Customer                          1      Customer
-balance:double = 0.0
------------------------------------
+Account(id:int,customer:Customer,
   balance:double)
+Account(id:int,customer:Customer)
+getID():int
+getCustomer():Customer                  "name(ID) balance=$xxx.xx"
+getBalance():double                     balance rounded to 2 decimal places
+setBalance(balance:double):void         To reuse Customer's toString()
+toString():String   •----------------
+getCustomerName():String
+deposit(amount:double):Account  •----   Add amount to balance
+withdraw(amount:double):Account •
```

if (balance >= amount)
   subtract amount to balance
else
   print "amount withdrawn
   exceeds the current balance!"

**Invoice**

-ID:int
-customer:Customer
-amount:double

+Invoice(ID:int,customer:Customer,
    amount:double)
+getID():int
+getCustomer():Customer
+setCustomer(customer:Customer):void
+getAmount():String
+setAmount(amount:double):void
+getCustomerName():String
+getAmountAfterDiscount():double

1    **Customer**

---

**Q2. Write a Java program to create MyPoint, MyCircle, and MyTriangle classes. Test the created classes by creating an object for each class.**

**MyPoint**

-x:int = 0
-y:int = 0

+MyPoint()
+MyPoint(x:int,y:int)
+getX():int
+setX(x:int):void
+getY():int
+setY(y:int):void
+getXY():int[2]
+setXY(x:int,y:int):void
+toString():String
+distance(x:int,y:int):double
+distance(another:MyPoint):double
+distance():double

Return a 2-element int array of {x,y}.

"(x,y)"

Distance from this point to the given point at (x,y).

Distance from this point to the given instance of MyPoint.

Distance from this point to (0,0).

**MyCircle**

```
-center:MyPoint = (0,0)
-radius:int = 1
```

```
+MyCircle()
+MyCircle(x:int,y:int,radius:int)
+MyCircle(center:MyPoint,radius:int)
+getRadius():int
+setRadius(radius:int):void
+getCenter():MyPoint
+setCenter(center:MyPoint):void
+getCenterX():int
+setCenterX(x:int):void
+getCenterY():int
+setCenterY(y:int):void
+getCenterXY():int[2]
+setCenterXY(x:int,y:int):void
+toString():String
+getArea():double
+getCircumference():double
+distance(another:MyCircle):double
```

1

**MyPoint**

```
-x:int
-y:int
```

center

"MyCircle[radius=r,center=(x,y)]"
Re-use MyPoint's toString() to print the center's (x,y).

Return the distance between the centers of this circle and the given MyCircle instance another. You need to re-use MyPoint's distance().

**MyTriangle**

```
-v1:MyPoint
-v2:MyPoint
-v3:MyPoint
```

```
+MyTriangle(x1:int,y1:int,
   x2:int,y2:int,x3:int,
   y3:int)
+MyTriangle(v1:MyPoint,
   v2:MyPoint,v3:MyPoint)
+toString():String
+getPerimeter():double
+getType():String
```

3

**MyPoint**

```
-x:int
-y:int
```

vertices

"MyTriangle[v1=(x1,y1),v2=(x2,y2),v3=(x3,y3)]"

Use MyPoint's distance() to compute the length of the edges.

"Equilateral" or "Isosceles" or "Scalene"