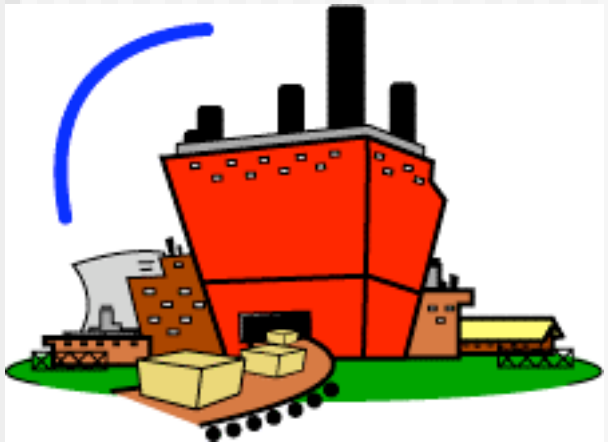
Computer Science 101 SS C

Lecture 16

Classes 3



Contents

- `this`
- `Multiple constructors`
- `null`
- `equals()` method
- Equality of objectsPractice defining a class - ForSale class

Coursebook: pages **§14.2.3 , §14.4.4, §14.4.6, §14.5**

The word, `this`

Sometimes an instance method defined in a class needs to refer to the object which is currently executing that instance method. The word, **`this`**, is used to refer to the current object i.e. the object which is currently executing the method.

The current object

item2 →

iDNumber	2315
description	"Dire Straits - Brothers in Arms LP"
contact	"027-453976"
askingPrice	56
isSold	false

item3 →

iDNumber	2314
description	"1976 Diary - Hardly used"
contact	"021-4563456"
askingPrice	23
isSold	false

Assume the two objects on the left have been created.
When the following code is executed:

```
item2.setAskingPrice(34);
```

`item2` is the current object.
When the following code is executed:

```
item3.setAskingPrice(47);
```

`item3` is the current object.

The word, `this`

Sometimes, inside a class, we need to refer to an instance variable belonging to the current object. We use the word, **`this`**, to refer to the current object. For example:

```
public class ForSale {
      private int iDNumber, askingPrice;
      private String description, contact;
      private boolean isSold;
      public ForSale(int iD, String phone, int
                                    price, String info) {
            this.iDNumber = iD;
            this.description = info;
            this.askingPrice = price;
            this.contact = phone;
            this.isSold = false;
      }
}
```

Example - The word, `this`

In the following example, **`this.description`** refers to the instance variable of the current object and **`description`** refers to the parameter.

```
public class ForSale {
      private int iDNumber, askingPrice;
      private String description, contact;
      private boolean isSold;
      public ForSale(int iDNumber, String contact,
                  int askingPrice, String description) {
            this.iDNumber = iDNumber;
            this.description = description ;
            this.askingPrice = askingPrice ;
            this.contact = contact ;
            this.isSold = false;
      }
}
```

Many constructors

A class can have more than one constructor. For example:

```
public class MarkAss1 {
    private int mark;
    private String uPI;
    public MarkAss1() {
        mark = 0;
    }
    public MarkAss1(String uPI) {
        this.uPI = uPI;
    }
    public MarkAss1(String uPI, int mark) {
      this.mark = mark;
      this.uPI = uPI;
    }
}
```

Ex03-Complete the constructor

```
public class MarkAss1 {
   private int mark;
   private String uPI;

   public MarkAss1(String uPI, int mark){




   }
}
```

Default values

Whenever an instance of a class is created all the instance variables are assigned their default value unless the constructor initialises them to a different value.

Default values for primitives

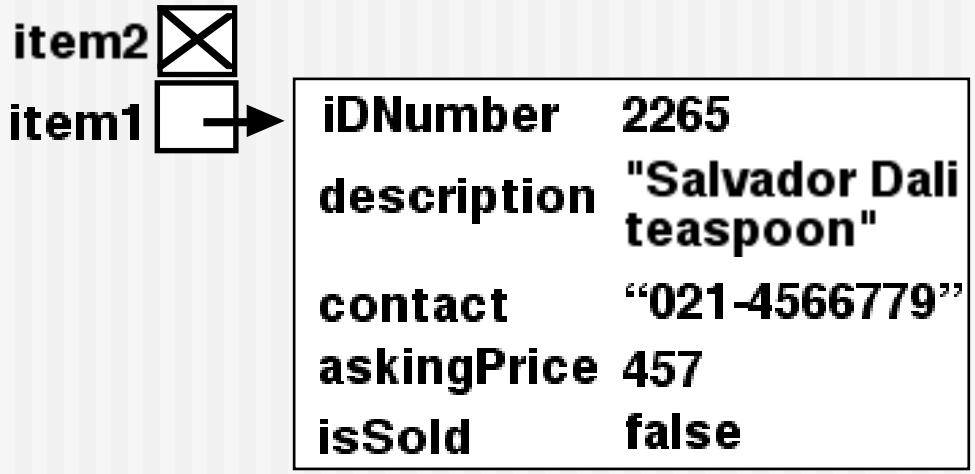
The default values for some of the primitive variables are:

Type	Value
int	0
double	0.0
boolean	false

The word, `null`

For an instance variable which is an object type the default value is **`null`**. **`null`** means that the variable does not point to an object yet.

```
ForSale item1, item2;
item1 = new ForSale(2265, "021-4566749",
                    457, "Salvador Dali teaspoon");
```



Ex04 -The word, `null`

`null` can be assigned to any reference variable. I can test if an object is equal to **`null`**.

```
public class ProgramClassL16 {
 public void start() {
     System.out.println(getFirst(null));
     System.out.println(getFirst("SK"));
 }
 private String getFirst(String name) {
   if(name==null || name.length()==0) {
          return null;
   }
   return name.substring(0,1);
}}
```

NullPointerException

Be very careful never to call a method on an object which has the value, **`null`**.

```
public class ProgramClassL16 {
   public void start() {
      String word = null;
      int length = word.length();
   }
}
```

`equals()` method

All class definitions should contain an **`equals()`** method.

We use the **`equals()`** method to test if two instances of the same class contain exactly the same information.

Example - `equals()` method

```
public class MarkSum {
  private String name, title;
  private int totalMarks;
  private boolean hasPassed;
     //...

   public boolean equals(MarkSum other) {
      if (totalMarks == other.totalMarks
            &&   hasPassed == other.hasPassed
            &&   name.equals(other.name)
            &&   title.equals(other.title)) {
         return true;
      }
      return false;
   }
}
```

`equals()` method parameter

In the **`equals()`** method , even though all the instance variables of the **`MarkSum`** class are defined as **`private`**, we can refer to:

```
other.name, other.hasPassed,
other.title, other.totalMarks
```

where **`other`** is a variable of type **`MarkSum`**.

This can be done inside the **`MarkSum`** class with any instance of the **`MarkSum`** class.

Ex08 - Define the `equals()` method

```
public class MarkAss1 {
   private int mark;
   private String uPI;
   public MarkAss1(String uPI,int mark) {
   }  ...
   public boolean equals(MarkAss1 other){




   }
}
```

Ex09 - What is the output?

```
MarkAss1  mark1, mark2, mark3;
mark1 = new MarkAss1("afer023", 97);
mark2 = mark1;
mark3 = new MarkAss1("afer023", 97);
if(  mark2 == mark3) {
     System.out.println("1.");
}else {
     System.out.println("2.");
}
if(mark2.equals(mark3)) {
     System.out.println("3.");
}else {
     System.out.println("4.");
}
```

Car for sale

The following is an actual advertisement in an Irish Newspaper..!

1985 Blue Volkswagen Golf Only 15 km
Only first gear and reverse used
Never driven hard Original tyres
Original brakes
Original fuel and oil
Only 1 driver
Owner Wishing to sell due to employment lay-off

`ForSale` - instance variables

Consider defining a class to represent items which are for sale on a website. Below is a visualisation of two typical items of the `ForSale` class.

iDNumber	**2315**
description	**"Dire Straits - Brothers in Arms LP"**
contact	**"027-453976"**
askingPrice	**56**
isSold	**false**

iDNumber	**2314**
description	**"1976 Diary - Hardly used"**
contact	**"021-4563456"**
askingPrice	**23**
isSold	**false**

`ForSale` - instance variables

The instance variables represent the information which all instances of the `ForSale` class will contain.

```
public class ForSale {
    private int iDNumber;
    private String description;
    private String contactDetails;
    private int askingPrice;
    private boolean isSold;

}
```

ForSale - constructor

```
public class ForSale {
      private int iDNumber;
      private String description;
      private String contact;
      private int askingPrice;
      private boolean isSold;
      public ForSale(int iD, String contact, int
                                        price, String info) {
            iDNumber = iD;
            description = info;
            askingPrice = price;
            contactDetails = contact;
            isSold = false;
}     }
```

`ForSale` - instance methods

Decide which operations?

`setIsSold()`: change the value stored in the boolean variable, isSold.

`getIsSold()`: obtain the value stored in the boolean variable, isSold.

`setAskingPrice()`: change the value stored in the int variable, askingPrice.

`getAskingPrice()`: obtain the value stored in the int variable, askingPrice.

`toString()`: obtain a String description of the object.

ForSale - instance methods

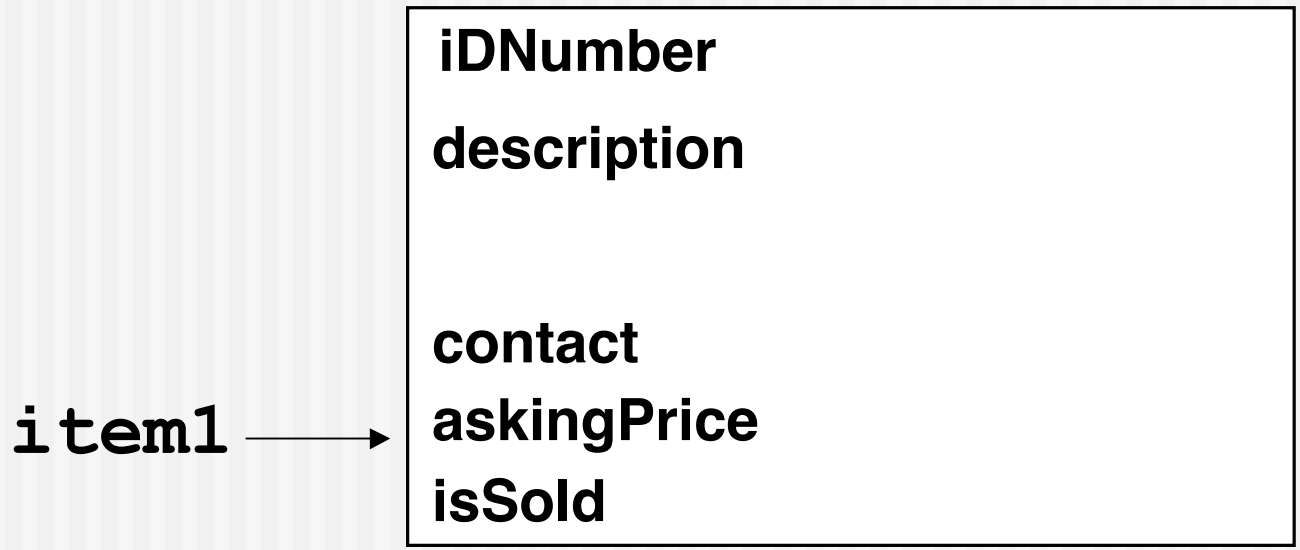
```
public class ForSale {
      private int iDNumber;
      private String description, contact;
      private int askingPrice;
      private boolean isSold;
      public ForSale(int iD, String phone,
                        int price, String info) {
            iDNumber = iD;
            description = info;
            askingPrice = price;
            contact = phone;
            isSold = false;
      }
      public boolean getIsSold() {
            return isSold;
      }
```

ForSale - instance methods

```
public void setIsSold(boolean sold) {
      isSold = sold;
}
public int getAskingPrice() {
      return askingPrice;
}
public void setAskingPrice(int price) {
      askingPrice = price;
}
public String toString() {
   String info = "Article: " + iDNumber;
   info = info + " " + description + ",";
   info = info + "\n Contact: " + contact;
   info = info + ", $" + askingPrice;
   if (isSold) {
     info = info + " SOLD";
   }
   return info;
}}
```

Ex01 - Draw the object

```
public class ProgramClassL16 {
  public void start() {
      ForSale item1;
      int idNumber = 4456;
      item1 = new ForSale(idNumber, "021-4566749", 457,
                                    "Salvador Dali teaspoon");
  }
}
```



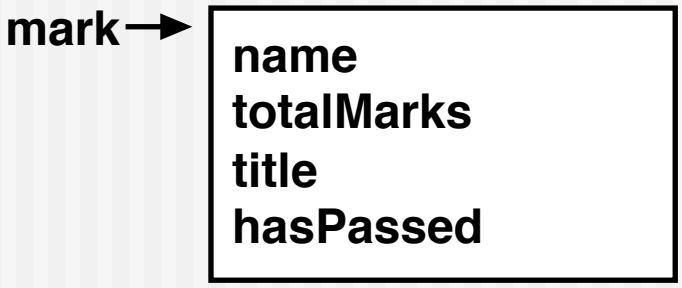
Ex02 - What is the output?

```
public class ProgramClassL16 {
 public void start() {
      int idNum = 4456;
      String info;
      ForSale item1 = new ForSale(idNum, "021-4566749",
                          457, "Salvador Dali teaspoon");

      item1.setAskingPrice(4079);
      item1.setIsSold(true);
      info = item1.toString();

      if (item1.getIsSold()) {
        System.out.println(info);
      } else {
        System.out.println(info + " is available");
      }
 }
}
```

Ex05 - Show the object, `mark`

```
public class MarkSum {
  private String name, title;
  private int totalMarks;
  private boolean hasPassed;
  public MarkSum(String name, String title) {
          this.name = name;
          this.title = title;
   }}
```

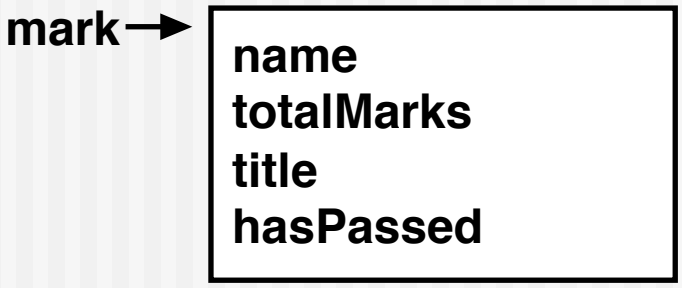
```
MarkSum mark = new MarkSum("Tom Li", "Mr");
```



Ex06 - Show the object, `mark`

```
public class MarkSum {
  private String name, title;
  private int totalMarks;
  private boolean hasPassed;
  public MarkSum(String name,int totMarks) {
          this.name = name;
          totalMarks = totMarks;
   }}
```

```
MarkSum mark = new MarkSum("Tom Li", 97);
```



Ex07-What is the output?

```
String s1 = new String("Hip");
String s2 = new String("Hip");
String s3 = s2;
if (s1 == s2) {
  System.out.println("1.");
}
if (s2 == s3) {
  System.out.println("2.");
}
if (s1.equals(s2)) {
  System.out.println("3.");
}
if (s2.equals(s3)) {
  System.out.println("4.");
}
```

Ex10 - Define `isHigherMark()`

```
public class MarkAss1 {
   private int mark;
   private String uPI;
   public MarkAss1(String uPI,int m){…}
   public boolean equals(MarkAss1 o){…}

   public boolean isHigherMark(MarkAss1
                                    other) {



} }
```

Ex11 - Give the output

```
public class ProgramClassL16 {
 public void start() {
      MarkAss1  mark1, mark2, mark3;
      mark1 = new MarkAss1("afer023", 98);
      mark2 = new MarkAss1("good005", 99);
      mark3 = new MarkAss1("vgd002", 98);
      if (mark2.isHigherMark(mark1)) {
         System.out.println("mark2 is better");
      }
      if (mark3.isHigherMark(mark1)) {
         System.out.println("mark3 is better");
      }
 }}
```

Ex12 - What is the problem?

```
public class ProgramClassL16 {
 public void start() {
    Chocolate choc;
    int price = choc.getPrice();
 }
}
```

Ex13 - What is the problem?

```
public class MarkAss1 {
    private int mark;
    private String uPI;

    public MarkAss1(String uPI, int mark) {
     this.mark = mark;
     this.uPI = uPI;
    }
    public MarkAss1(String u, int m) {
     mark = m;
     uPI = u;
    }
}
```

What you need to know

Structure of a class definition
- Instance variables
- Constructors
- Instance methods

Using **`this`** in the constructor.

Object variables have the default value, **`null`**.

Defining an **`equals()`** method.

`==` vs **`equals()`**