Object Orientation with Design Patterns

Lecture 2 : Constructors - Revision

The Role of Constructors

- As we have seen from previous examples classes can have set & get methods which allow the programmer to access the private data members encapsulated within the class.
- In all our previous examples we created an object variable, allocated space for the object using the new operator and then called the set and get methods on the object.

```
public static void main(String[] args)
{
    // Create 2 object variables of
    // type CompactDisc
    CompactDisc cd1;
    CompactDisc cd2;

    // Allocate some memory for
    // each new object
    cd1 = new CompactDisc();
    cd2 = new CompactDisc();

    cd1.setTitle("Kid A");
```

The Role of Constructors

- What if we wanted to use an object **which when created was also initialised with some default values.** So instead of creating the object and then calling its set methods all we would have to do is create it and the initialisation would happen automatically.

- In order to achieve this all we have to do is write a special type of method for our class which will do the initialisation for us. **These special methods are called *constructors***.

- There are a couple of important points to remember when writing constructors.
  - **A constructor must have the same name as the class**
  - **A constructor cannot have a return type declared as it is not allowed to return any values (not even void).**

The Role of Constructors

- For example if we were to write a class that modelled an integer counter we may want the **internal counter variable to be initialised to zero** when an object of that type was created.

```
class Counter
{
    private int count;

    public Counter()
    {
        count = 0;
    }
    public void setCount(int val)
    {
        count = val;
    }
    public int getCount()
    {
        return count;
    }
    public void Inc()
    {
        count++;
    }
    public void Dec()
    {
        count--;
    }
}
```

Constructor

The Role of Constructors

- If we were to create an object of type **Counter the constructor would be called automatically and the internal data member, *count,* would be initialised to zero.**

- So constructors are not **explicitly called**. If a class has a constructor method it **will be called automatically** once the new operator has allocated memory for the object.

```
Counter c1;
c1 = new Counter();  // Constructor gets called here
```

- So if a class has a constructor method you can be guaranteed that it will be the **first method** to be called on the object. The next slide shows a test program for the counter class.

The Role of Constructors

```
public static void main(String[] args)
{
    // declare two counter object variables
    Counter c1, c2;

    // allocate memory for the object
    // using the new operator
    c1 = new Counter();
    c2 = new Counter();

    // Test the value of the counter objects
    Screen.message("Values after construction");
    Screen.newline();
    Screen.message("Counter 1 = " + c1.getCount());
    Screen.newline();
    Screen.message("Counter 2 = " + c2.getCount());
    Screen.newline();

    // Test the Inc & Dec methods
    c1.Inc();
    c2.Dec();

    // Test the value of the counter objects
    Screen.message("Values after c1.Inc() & c2.Dec()");
    Screen.newline();
    Screen.message("Counter 1 = " + c1.getCount());
    Screen.newline();
    Screen.message("Counter 2 = " + c2.getCount());
    Screen.newline();
}
```

Constructors get called

The Role of Constructors

- If we examine the output of the counter test program we can see that the two counter objects are initialised to zero as soon as the new operator is executed.

```
C:\WINNT\System32\CMD.exe
Values after construction
Counter 1 = 0
Counter 2 = 0
Values after c1.Inc() & c2.Dec()
Counter 1 = 1
Counter 2 = -1
Press any key to continue . . .
```

The Role of Constructors

- As well as initialising internal data members we can **also call other methods** from within a constructor.

```
public Counter()
{
    // Use the set method instead of writing the
    // assignment statement again
    setCount(0);

    // Call a method on a different object
    Screen.message(" I'm alive !");
    Screen.newline();
}
```

```
C:\WINNT\System32\CMD.exe
 I'm alive !
 I'm alive !
Values after construction
Counter 1 = 0
Counter 2 = 0
Values after c1.Inc() & c2.Dec()
Counter 1 = 1
Counter 2 = -1
Press any key to continue . . .
```

The Role of Constructors

- The constructors we have looked at so far are known as default constructors. They are called ***default* constructors** because they **do not take any parameters**.

- There is however another type of constructor, one **which does take parameters**. This type of constructor is called a ***user defined* constructor.**

- So for example we could have written a constructor for the counter class which took as its parameter an integer value which could be used to initialise the *count* data member.

- This means we would no longer have to hard code the initial value in the constructor code because the programmer could pass a value in to the constructor during the call to *new.*

The Role of Constructors

- The following code shows the counter class with a user defined constructor that takes as its input an integer value.

```
class Counter
{
    private int count;

    public Counter(int val)
    {
        count = val;
    }
    public void setCount(int val)
    {
        count = val;
    }
    public int getCount()
    {
        return count;
    }
    public void Inc()
    {
        count++;
    }
    public void Dec()
    {
        count--;
    }
```

User defined constructor

The Role of Constructors

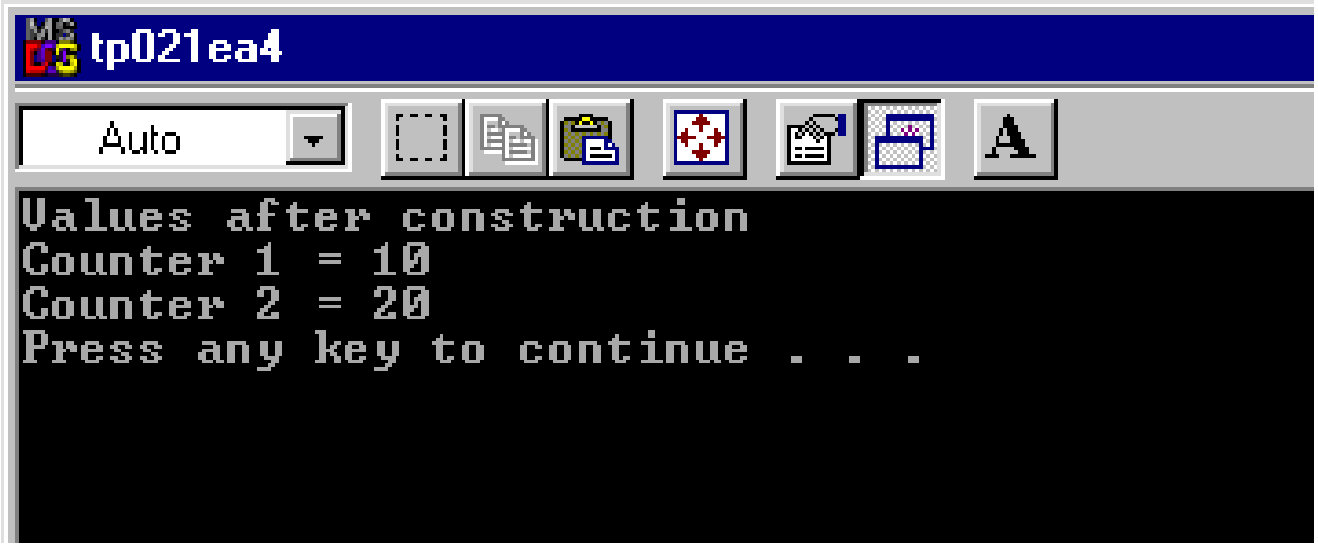
- We can test the new user defined constructor with the following code.

```
public static void main(String[] args)
{
    Counter c1,c2;

    c1 = new Counter(10);
    c2 = new Counter(20);

    Screen.message("Values after construction");
    Screen.newline();
    Screen.message("Counter 1 = " + c1.getCount());
    Screen.newline();
    Screen.message("Counter 2 = " + c2.getCount());
}
```

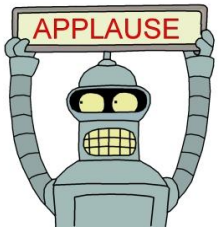
Pass in an initial value while creating the object

The Role of Constructors

- When the test program is executed we get the following output.



Summary



- We have seen how constructors are used to initialise an objects data members, either by using a **default constructor** where the values are hard coded or by using a **user defined constructor** where values can be passed.