Reference Variables

Elaine Tynan

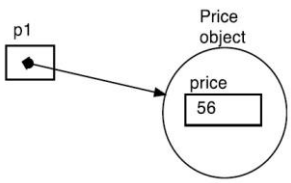
Reference Variables

- A Reference Variable points to an object (Not a primitive type)

```
class Price{
  private int price;
  public Price(){ price = 56;}
  public int price(){ return price;}
  public void setPrice(int k){ price = k;}
  public void display(){
    System.out.println("Price: "+ price);
  }
}
```

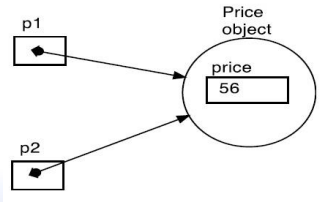
Reference Variables

Price p1 = **new Price();**



Reference Variables

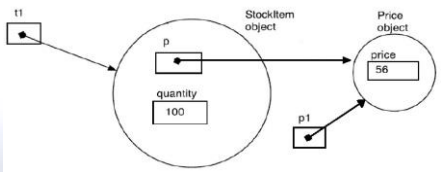
```
Price p1 = new Price();
Price p2 = p1;
```



```
class StockItem{
  private int quantity;
  private Price p;
  StockItem(int q, Price p){
    quantity = q; this.p= p;
  }
  public Price price(){ return p; }
  public void setItemPrice(int k) {
    p.setPrice(k);
  }
  public void display(){
    System.out.println(quantity+" "+p.price());
  }
}
```

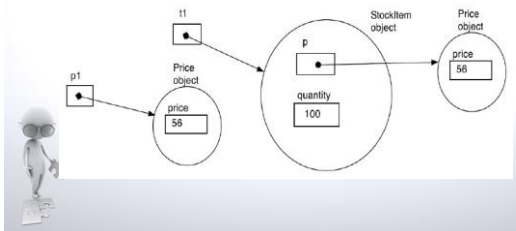
Price p1 = new Price();

StockItem t1 = new StockItem(100, p1);



```
class StockItem{
  private int quantity;
  private Price p;
  StockItem(int q, Price p){
    quantity = q; this.p= new Price();
    this.p.setPrice(p.price());
  }
  public int price(){ return p.price(); }
  public void setItemPrice(int k){
    p.setPrice(k);
  }
  public void display(){
    System.out.println(quantity+“”+p.price());
  }
}
```

```
Price p1 = new Price();
StockItem t1 = new StockItem(100, p1);
```



Another Example

```
class Member{
  private String name;  private Date born;
  public Member(String n, int d, int m, int y){
    name = n;
    born = new Date(y,m,d);
  }
  public String dob(){
    return born.getDate()+ “-” +born.getMonth()
                  + “-” + born.getYear();
  }
  public void display(){
    System.out.println("Name: " + name);
    born.display();
  }
}
```

Data Encapsulation

- Encapsulation in Java is a mechanism of wrapping the data (variables) and code acting on the data (methods) together as a single unit
- In encapsulation, the variables of a class will be hidden from other classes, and can be accessed only through the methods of their current class
- It is also known as **data hiding**

Data Encapsulation

- To achieve encapsulation in Java
  - Declare the variables of a class as private
  - Provide public setter and getter methods (where appropriate) to modify and view the variables values

Mutable Objects

- **mutable objects** are objects whose state can be changed after construction
- They should be written so that encapsulation is never broken
- The state should only be changed by the class itself

Mutable Objects

- In the event where one of these fields is returned by a method then a new copy (called a defensive copy) of the mutable object must be made and returned
- This rule holds true for constructors, set methods and get methods

```
class Counter{
   private int count;
   Counter(){ count = 0; }
   Counter(int k){ count = k; }
   public void inc(){ count++; }
   public int count(){ return count; }

   public static void main(String args){
      Counter c = new Counter();
      c.inc();
      //Can't go c++;
   }
}
```

Objects Passed by Reference

```
class Car{
   private String name;
   private Counter odometer;
   Car(String n, Counter c){
      name = n;
       odometer = new Counter(c.count());
   }
   public String name(){return name;}
   public Counter kilometre(){
      return new Counter(odometer.count());
   }
   public int getDistance(){
      return odometer.count();
   }
   public void inc(){odometer.inc();}
}
```

In main

```
Car cr1 = new Car("Honda",
                  new Counter());

Counter c1 = new Counter(1000);
Car cr2 = new Car("Ford", c1 );
cr1.inc();
```

```
class Member {
   private String name;
   private Date born;
   public Member(String n, int d, int m, int y) {
      name = n;
      born = new Date(y,m,d);
   }
   public String dob() {
      String db = born.getDate() +"/"+ born.getMonth()
                                  +"/"+ born.getYear();
      return db;
   }
   public void display() {
      System.out.println("Name: " + name);
      System.out.println(Date of Birth: "+born.toString());
   }
}
```

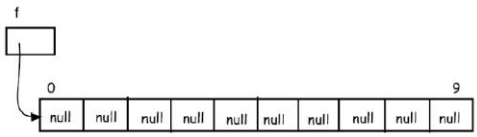
Declaring an Array

Arrays Of Objects

There are two steps involved.

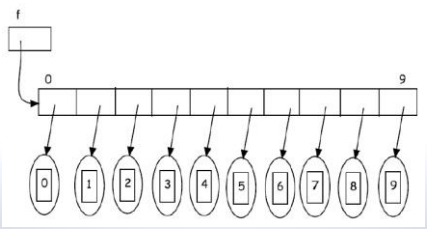
1. Declare the array. The format is:
   className[] name = new className[size];
   For example:
   Integer[] f = new Integer[10];
2. Create instances of the object className and assign their reference to the array element.

Arrays Of Objects



Arrays Of Objects

```
for(int j = 0; j < f.length; j++)
      f[j] = new Integer(j);
```



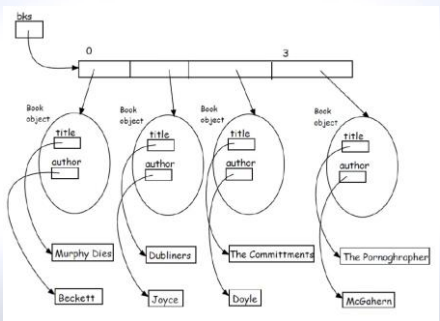
Create an Arrays Of Strings

```
String[] f1 = {“Malone Dies”, “Murphy Dies”,
            “EndGame”, “Waiting for Godot”};
System.out.println();
for (String s : f1)
    System.out.println(s);
```

Create an Arrays Of Books

```
Book[] bks = {
      new Book(“Murphy Dies”, “Beckett”),
      new Book(“Dubliners”, “Joyce”),
      new Book(“The Commitments”, “Doyle”),
      new Book(“Room”, “Donoghue”)
};
System.out.println();
for (Book b : bks)
    System.out.println(b.toString() );
```

Create an Arrays Of Books



```
class Library{
  private Book[] books;
  private int size;
  public Library(int n){
    size = 0;
    books = new Book[n];
  }
  public void add(String t, String a) {
    if (size < books.length) {
      books[size] = new Book(t, a);
      size++;
    }
  }
```

```
public boolean full() {
  return (size==books.length);
}

public int numBooks() {
  return size;
}
```

```
public Book searchTitle(String t){
  boolean found = false;
  int j = 0;
  while(j < size && !found){
    if(t.equals(books[j].title()))
      found = true;
    else
      j++;
  }
  if(found) return books[j];
  else return null;
}
```

```
  public void display(){
    for(int j = 0; j < size; j++)
      books[j].display();
  }
}
```

```
public class LibraryTest {
  public static void main(String[] args) {
    Library l = new Library(100); // Library of 100 books
    l.add("Murphy Dies", "Beckett");
    l.add("Dubliners", "Joyce")
    System.out.println(l.numBooks() );
    Book b = l.searchTitle("Dubliners");
    if(b!=null)
      b.display();
    l.display();
  }
}
```

```
public Book[] searchByAuthor(String a){
  // To return a list of books by a given author 1st we
  // get the frequency of books by the author
  // and then return the books
  int freq = 0;
  for(int j = 0; j < size; j++){
    if(a.equals(books[j].author())) freq++;
  }
  Book[] bks = new Book[freq];
  int k = 0;
  for(int j = 0; j < size; j++){
    if(a.equals(books[j].author())){
      bks[k] = books[j];
      k++;
    }
  }
  return bks;
}
```

```
public class LibraryTest {
  public static void main(String[] args) {
    Library l = new Library(100); // Library of 100 books
    …
    Book[] bks = l.searchByAuthor("Doyle");
    for(Book b : bks)
      System.out.println(b.toString() );

    // Below does the same as above loop
    for(int i=0; i<bks.length; i++) {
      System.out.println(bks[i]);
    }
  }
}
```

elaine.tynan@griffith.ie