Programming Fundamentals 2

Introduction to ArrayList (contd.)

Produced by: Dr. Siobhán Drohan
Mairead Meagher

Waterford Institute *of* Technology
INSTITIÚID TEICNEOLAÍOCHTA PHORT LÁIRGE

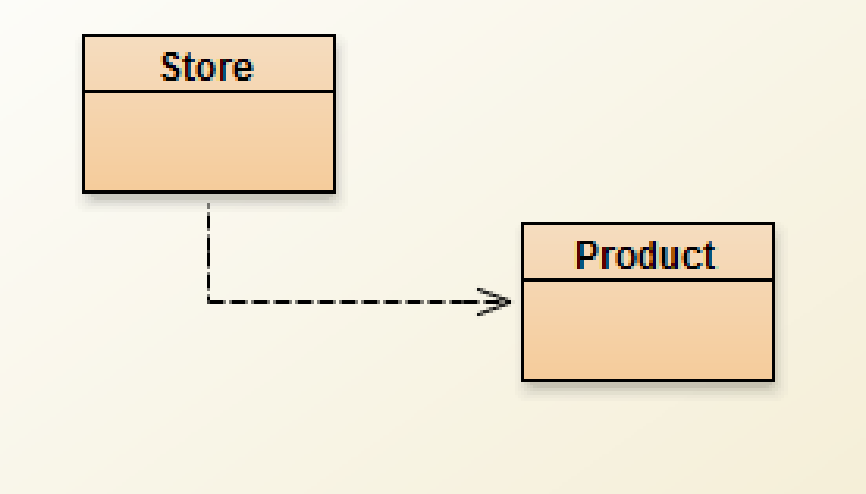
Department of Computing and Mathematics
http://www.wit.ie/

Topic list

- Grouping Objects
  - Developing a basic personal notebook project using Collections e.g. ArrayList
- Indexing within Collections
  - Retrieval and removal of objects
- Generic classes e.g. ArrayList
- Iteration
  - Using the for each loop
  - Using the while loop
- Coding ShopV3.0 that stores an ArrayList of Products.

A basic example of a Shop

A Store has an ArrayList of Product.



Product class

- Our Product class contains four instance variables (requiring no validation at the moment).
- The class has a constructor that uses the data passed in the four parameters to update the instance fields.

```
public Product(String productName, int productCode, double unitCost, boolean inCurrentProductLine)
{
    this.productName = productName;
    this.productCode = productCode;
    this.unitCost = unitCost;
    this.inCurrentProductLine = inCurrentProductLine;
}
```

- The class has getters and setters for each instance field.
- It also has a toString() method.

Store class

- The Store class will contain:
  1. an ArrayList of Product.
  2. a method to add Products to the ArrayList.
  3. a method to print out the contents of the ArrayList.
  4. a method that will print out the cheapest product in the ArrayList.

Store class

- The Store class will contain:
  1. an ArrayList of Product.
  2. a method to add Products to the ArrayList.
  3. a method to print out the contents of the ArrayList.
  4. a method that will print out the cheapest product in the ArrayList.

1. Declaring an ArrayList of Product

```
import java.util.ArrayList;

public class Store
{
        private ArrayList<Product> products;

        // constructor
        public Store()
        {
            products = new ArrayList<Product> ();
        }

}
```

1. Declaring an ArrayList of Product

importing the ArrayList class so we can use it.

declaring an ArrayList of Product as a private instance variable.

calling the constructor of the ArrayList class to build the ArrayList object.

```
import java.util.ArrayList;

public class Store
{
        private ArrayList<Product> products;

        // constructor
        public Store()
        {
        products = new ArrayList<Product> ();
        }

}
```

Store class

- The Store class will contain:
  1. an ArrayList of Product.
  2. a method to add Products to the ArrayList.
  3. a method to print out the contents of the ArrayList.
  4. a method that will print out the cheapest product in the ArrayList.

2. Add a Product object to an ArrayList of Product

```
public void add (Product product)
{
        products.add (product);
}
```

add: This is add method from the ArrayList class that we just imported.

products: This is the ArrayList of Product.

Product: The ArrayList holds objects of this type, Product.

product: This is object of type Product that we want to add to the ArrayList.

2. Add a Product object to an ArrayList of Product

```
import java.util.ArrayList;

public class Store{

        private ArrayList<Product> products;

        public Store(){
            products = new ArrayList<Product> ();
        }

        public void add (Product product){
            products.add (product);
        }
}
```

Store class

- The Store class will contain:
  1. an ArrayList of Product.
  2. a method to add Products to the ArrayList.
  3. a method to print out the contents of the ArrayList.
  4. a method that will print out the cheapest product in the ArrayList.

3. Printing all Products in an ArrayList of Product

```
public void listProducts(){
        for (Product product: products){
            System.out.println(product.toString());
        }
}
```

Product: This is the type of object that is stored in the ArrayList.

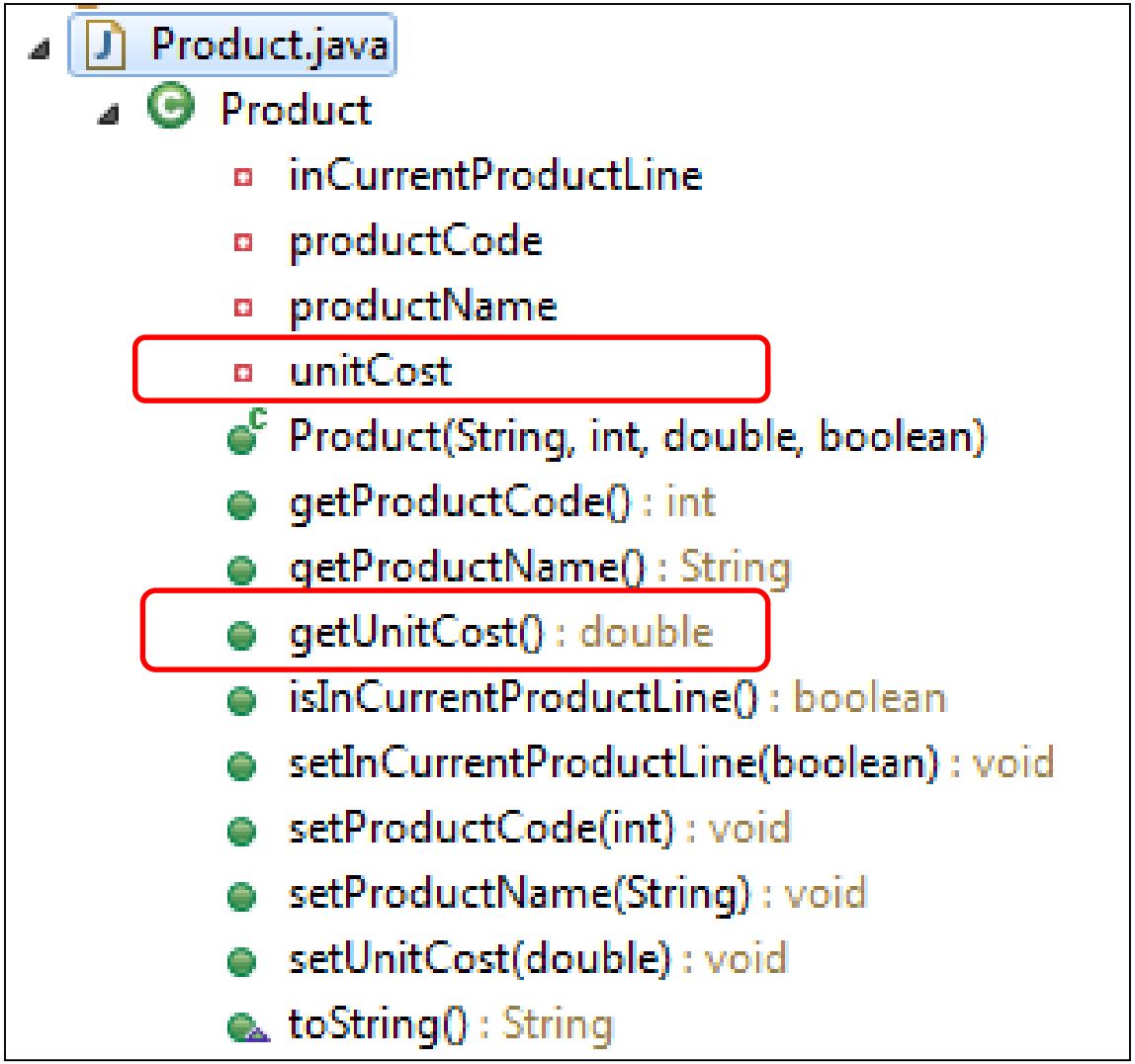
product: This is object reference pointing to the current object we are looking at in the ArrayList. As we iterate over each object in the ArrayList, this reference will change to point to the next object, and so on.

products: This is the ArrayList of Product.

Store class

- The Store class will contain:
  1. an ArrayList of Product.
  2. a method to add Products to the ArrayList.
  3. a method to print out the contents of the ArrayList.
  4. a method that will print out the cheapest product in the ArrayList.

Finding the Cheapest Product



Finding the Cheapest Product

1. If products have been added to the ArrayList
   1.1 Assume that the first Product in the ArrayList is the cheapest (set a local variable to store this object).
   1.2 For all product objects in the ArrayList
      1.2.1 if the current product cost is lower than the cost of the product object stored in the local variable,
         1.2.1.1 update the local variable to hold the current product object.
      end if
   end for
   1.3 Return the name of the cheapest product.
   else
   1.4 Return a message indicating that no products exist.
   end if

Finding the Cheapest Product

Working on the outer if statement

```
if products have been added to the ArrayList
        // return the cheapest product
else
        return a message indicating that no products exist.
end if
```

How do we write the code for this algorithm?

Finding the Cheapest Product

Working on the outer if statement

```
if products have been added to the ArrayList
        //return the cheapest product
else
        return a message indicating that no products exist.
end if
```

```
if (products.size() > 0){
        //return the cheapest product
}
else{
        return “No products added to the ArrayList”;
}
```

Working on step 1.1

```
if products have been added to the ArrayList
        // 1.1   Assume that the first Product in the ArrayList is the
        // cheapest (set a local variable to store this object).
else
        return a message indicating that no products exist.
end if
```

How do we write the code for this step?

Working on step 1.1

```
if products have been added to the ArrayList
        // 1.1   Assume that the first Product in the ArrayList is the
        // cheapest (set a local variable to store this object).
else
        return a message indicating that no products exist.
end if
```

```
if (products.size() > 0){
        Product cheapestProduct = products.get(0);
 }
else{
        return “No products added to the ArrayList”;
}
```

Working on the for loop

```
if products have been added to the ArrayList
        // 1.1   Assume that the first Product in the ArrayList is the
        // cheapest (set a local variable to store this object).
        // 1.2   For all product objects in the ArrayList
        //         determine the cheapest product
        //  end for
else
        return a message indicating that no products exist.
end if
```

How do we write the code for this step?

Working on the for loop

```
if (products.size() > 0){
        Product cheapestProduct = products.get(0);
        for (Product product : products)
        {
        }
 }
else{
        return “No products added to the ArrayList”;
}
```

Working on the code inside the for loop

1. If products have been added to the ArrayList
   1.1 Assume that the first Product in the ArrayList is the cheapest (set a local variable to store this object).
   1.2 For all product objects in the ArrayList
      1.2.1 if the current product cost is lower than the cost of the product object stored in the local variable,
         1.2.1.1 update the local variable to hold the current product object.
      end if
   end for
   1.3 Return the name of the cheapest product.
   else
   1.4 Return a message indicating that no products exist.
   end if

How do we write the code for this?

Working on the code inside the for loop

```
if (products.size() > 0){
        Product cheapestProduct = products.get(0);
        for (Product product : products){
            if (product.getUnitCost() < cheapestProduct.getUnitCost() )
            {
            }
        }
 }
else{
        return “No products added to the ArrayList”;
}
```

Working on the code inside the for loop

1. If products have been added to the ArrayList
   1.1 Assume that the first Product in the ArrayList is the cheapest (set a local variable to store this object).
   1.2 For all product objects in the ArrayList
      1.2.1 if the current product cost is lower than the cost of the product object stored in the local variable,
         1.2.1.1 update the local variable to hold the current product object.
      end if
   end for
   1.3 Return the name of the cheapest product.
   else
   1.4 Return a message indicating that no products exist.
   end if

How do we write the code for this step?

Working on the code inside the for loop

```
if (products.size() > 0){
        Product cheapestProduct = products.get(0);
        for (Product product : products){
            if (product.getUnitCost() < cheapestProduct.getUnitCost() ){
                cheapestProduct = product;
            }
        }
 }
else{
        return “No products added to the ArrayList”;
}
```

Working on the last step, 1.3

1. If products have been added to the ArrayList
   1.1 Assume that the first Product in the ArrayList is the cheapest (set a local variable to store this object).
   1.2 For all product objects in the ArrayList
      1.2.1 if the current product cost is lower than the cost of the product object stored in the local variable,
         1.2.1.1 update the local variable to hold the current product object.
      end if
   end for
   1.3 Return the name of the cheapest product.
   else
   1.4 Return a message indicating that no products exist.
   end if

How do we write the code for this step?

Working on the last step, 1.3

```
if (products.size() > 0){
        Product cheapestProduct = products.get(0);
        for (Product product : products){
            if (product.getUnitCost() < cheapestProduct.getUnitCost()){
                cheapestProduct = product;
             }
        }
        return cheapestProduct.getProductName();
 }
else{
        return “No products added to the ArrayList”;
}
```

Review

- Collections allow an arbitrary number of objects to be stored.
- Class libraries usually contain tried-and-tested collection classes.
- Java's class libraries are called *packages*.
- We have used the `ArrayList` class from the `java.util` package.

Review

- Items may be added and removed.
- Each item has an index.
- Index values may change if items are removed (or further items added).
- The main `ArrayList` methods are `add`, `get`, `remove` and `size`.
- `ArrayList` is a parameterized or generic type.

Review

- Loop statements allow a block of statements to be repeated.
- The for-each loop allows iteration over a whole collection.
- The while loop allows the repetition to be controlled by a boolean expression.

Waterford Institute *of* Technology
INSTITIÚID TEICNEOLAÍOCHTA PHORT LÁIRGE

Department of Computing and Mathematics
http://www.wit.ie/