Sorting and Searching

ShopV8.0 - Basic sorting and searching algorithms

Produced by:

Dr. Siobhán Drohan
Ms. Mairead Meagher
Ms. Siobhan Roche

Ollscoil Teicneolaíochta an Oirdheiscirt
South East Technological University

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

Brief overview of Searching and Sorting

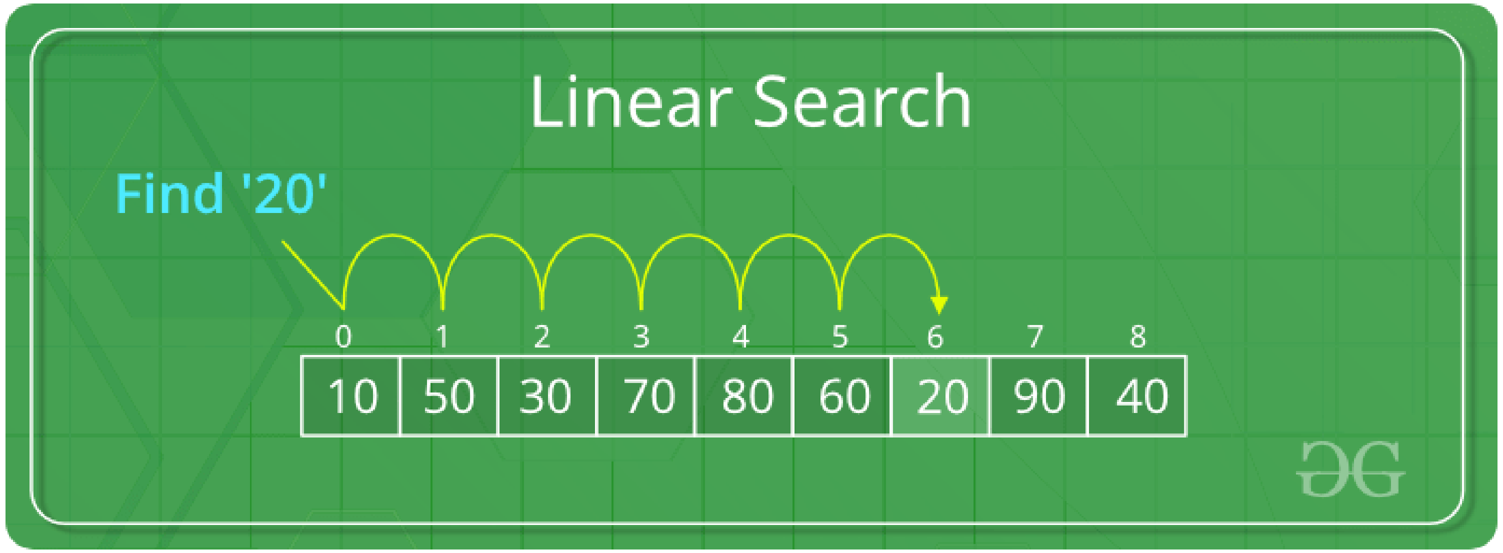
- Searching
  - Linear Search (unsorted list)
- ArrayList
  - Swapping Values
- Sorting
  - Selection Sort

Linear searching

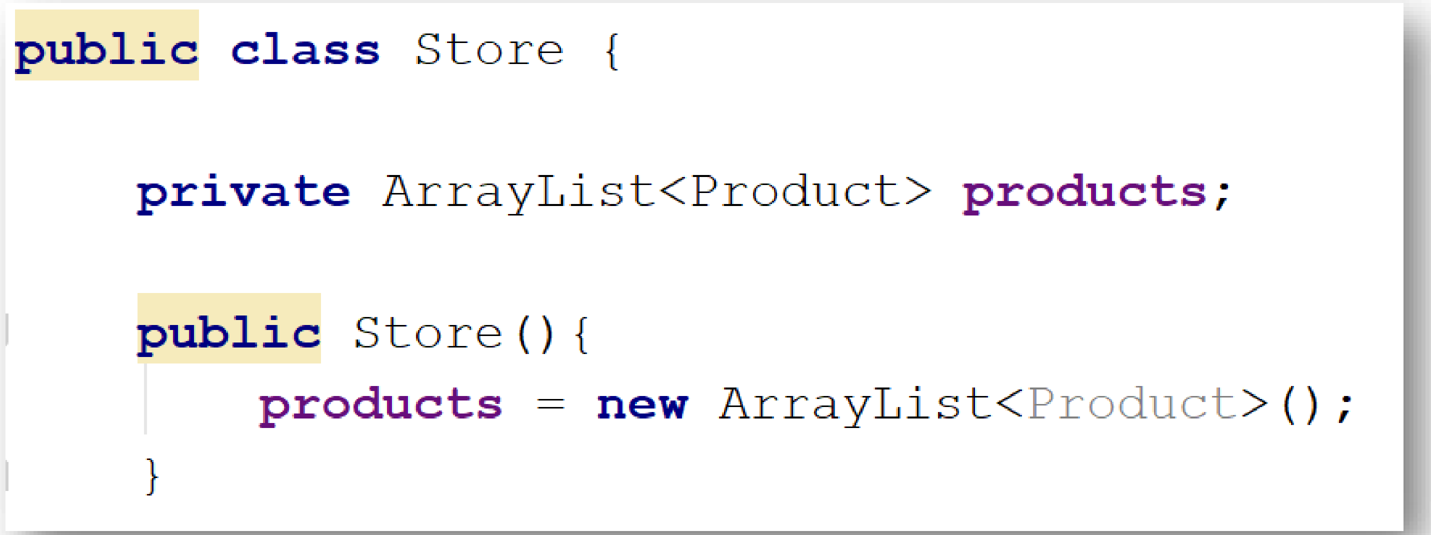
Searching

- Before we decide on how to search through a list we need to know whether or not the list is sorted.
  - Not Sorted → Linear Search
  - Sorted → Binary Search

Linear Search



ShopV8.0 – Linear Search



```
public class Store {

    private ArrayList<Product> products;

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

Problem: We want to search the **products** ArrayList and return all the individual products whose product name matches, or partially matches the search word entered by the user.

Store.java

```
public String searchByProductName(String productName)  {

     String matchingProducts = "";

     //todo: search for the matching product(s)

     if (matchingProducts.equals("")){
          return "No products match your search";
     }
     else{
          return matchingProducts;
     }
}
```

Problem: We want to search the **products** ArrayList and return all the individual products whose product name matches, or partially matches the search word entered by the user.

Store.java

```
public String searchByProductName(String productName) {

    String matchingProducts = "";


    for(Product product : products) {
        if (//todo: search criteria) {
            matchingProducts += product + "\n";
        }
    }

    if (matchingProducts.equals("")){
        return "No products match your search";
    }
    else{
        return matchingProducts;
    }
}
```

Problem: We want to search the **products** ArrayList and return all the individual products whose product name matches, or partially matches the search word entered by the user.

Store.java

```java
public String searchByProductName(String productName) {

    String matchingProducts = "";


    for(Product product : products) {
        if (product.getProductName().contains(productName)){
            matchingProducts += product + "\n";
        }
    }

    if (matchingProducts.equals("")){
        return "No products match your search";
    }
    else{
        return matchingProducts;
    }
}
```

Problem: We want to search the **products** ArrayList and return all the individual products whose product name matches, or partially matches the search word entered by the user.

Store.java

```
public String searchByProductName(String productName) {

    String matchingProducts = "";

    for(Product product : products) {
        if (product.getProductName().toUpperCase().contains(productName.toUpperCase())) {
            matchingProducts += product + "\n";
        }
    }

    if (matchingProducts.equals("")){
        return "No products match your search";
    }
    else{
        return matchingProducts;
    }
}
```

Catering for case sensitivity

Catering for case sensitivity

Problem: We want to search the **products** ArrayList and return all the individual products whose product name matches, or partially matches the search word entered by the user.

Store.java

```
public String searchByProductName(String productName) {

    String matchingProducts = "";


    for(Product product : products) {
        if (product.getProductName().toUpperCase().contains(productName.toUpperCase())) {
            matchingProducts += products.indexOf(product) + ": " + product + "\n";
        }
    }

    if (matchingProducts.equals("")){
        return "No products match your search";
    }
    else{
        return matchingProducts;
    }
}
```

Adding the index number for each product

Problem: We want to search the **products** ArrayList and return all the individual products whose product name matches, or partially matches the search word entered by the user.

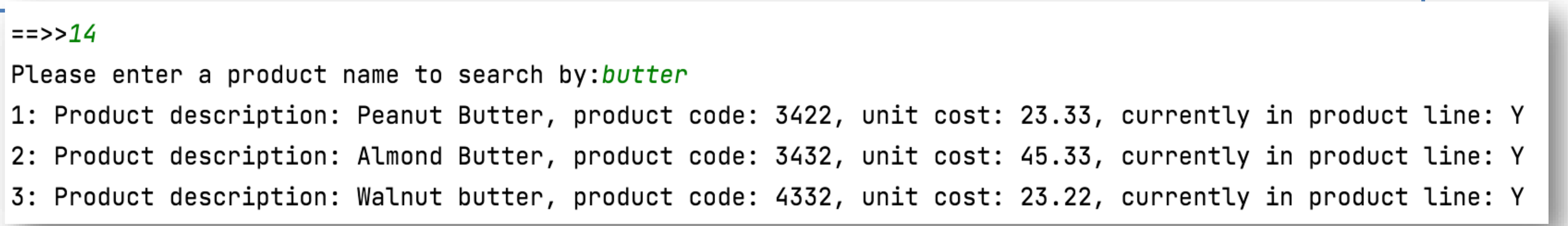
Driver.java

```java
private int mainMenu() {
    int option = ScannerInput.readNextInt("""
           -------------------------------------------------------------------
           |                          Shop Menu                              |
           -------------------------------------------------------------------
           | PRODUCT MENU                                                    |
           |   1) Add a product                                              |
           |   2) List the Products                                          |
           |   3) Update a product                                           |
           |   4) Delete a product                                           |
           -------------------------------------------------------------------
           | PRODUCT DESCRIPTION MENU                                        |
           |   5) Add a product description                                  |
           |   6) List product description(s)                                |
           |   7) Update a product description                               |
           |   8) Delete a product description                               |
           -------------------------------------------------------------------
           | REPORT MENU                                                     |
           |   10) List the current products                                 |
           |   11) Display average product unit cost                         |
           |   12) Display cheapest product                                  |
           |   13) List products that are more expensive than a given price |
           -------------------------------------------------------------------
           | SEARCH AND SORT MENU                                            |
           |   14) Search products by name                                   |
           |   15) Sort products by name ascending                           |
           -------------------------------------------------------------------
           |   20) Save products to products.xml                             |
           |   21) Load products from products.xml                           |
           |   0)  Exit                                                      |
           -------------------------------------------------------------------
           ==>>  """);
    return option;
}
```

```java
switch (option) {
    case 1 -> addProduct();
    case 2 -> printProducts();
    case 3 -> updateProduct();
    case 4 -> deleteProduct();
    case 5 -> addDescriptionToProduct();
    case 6 -> listProductDescriptions();
    case 7 -> updateDescriptionInProduct();
    case 8 -> deleteDescriptionFromProduct();
    case 10 -> printCurrentProducts();
    case 11 -> printAverageProductPrice();
    case 12 -> printCheapestProduct();
    case 13 -> printProductsAboveAPrice()
    case 14 -> searchProductByName();
    case 15 -> sortProductByName();
    case 20 -> saveProducts();
    case 21 -> loadProducts();
    default -> System.out.println("Invalid option entered: " + option);
}
```

Driver.java

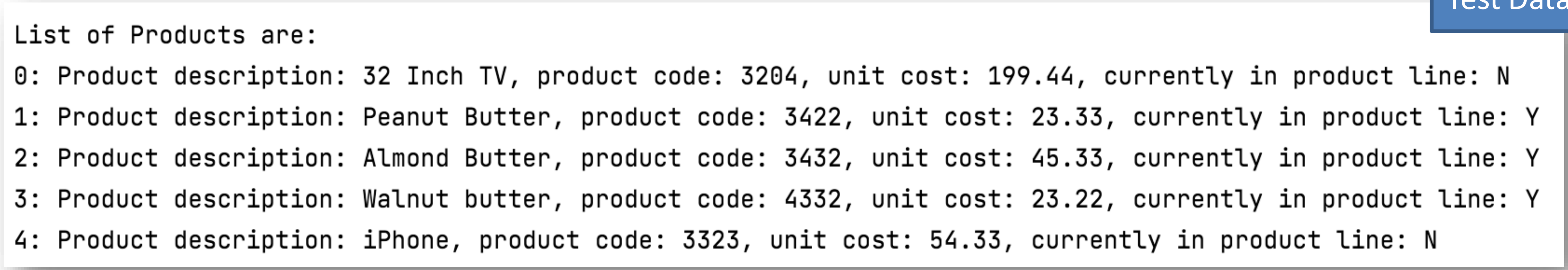
```java
private void searchProductByName()  {
    String productName = ScannerInput.readNextLine("Please enter a product name to search by:");
    System.out.println(store.searchByProductName(productName));
}
```



```
==>>14
Please enter a product name to search by:butter
1: Product description: Peanut Butter, product code: 3422, unit cost: 23.33, currently in product line: Y
2: Product description: Almond Butter, product code: 3432, unit cost: 45.33, currently in product line: Y
3: Product description: Walnut butter, product code: 4332, unit cost: 23.22, currently in product line: Y
```

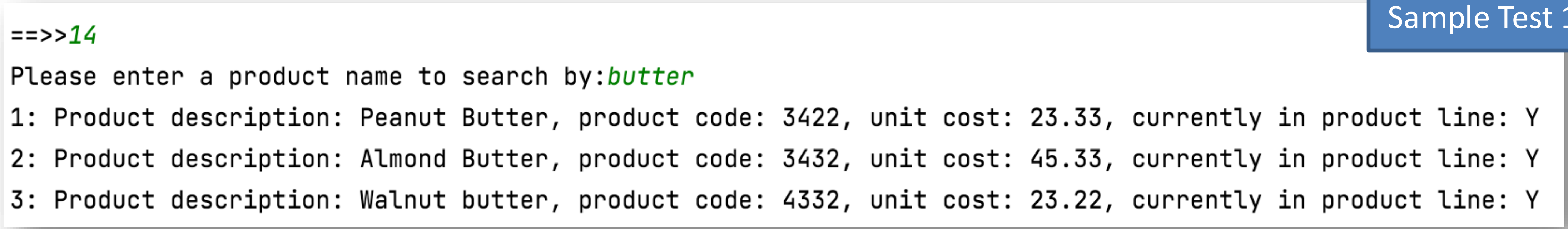
```java
switch (option) {
    case 1 -> addProduct();
    case 2 -> printProducts();
    case 3 -> updateProduct();
    case 4 -> deleteProduct();
    case 5 -> addDescriptionToProduct();
    case 6 -> listProductDescriptions();
    case 7 -> updateDescriptionInProduct();
    case 8 -> deleteDescriptionFromProduct();
    case 10 -> printCurrentProducts();
    case 11 -> printAverageProductPrice();
    case 12 -> printCheapestProduct();
    case 13 -> printProductsAboveAPrice();
    case 14 -> searchProductByName();
    case 15 -> sortProductByName();
    case 20 -> saveProducts();
    case 21 -> loadProducts();
    default -> System.out.println("Invalid option entered:  " + option);
}
```

Test Data



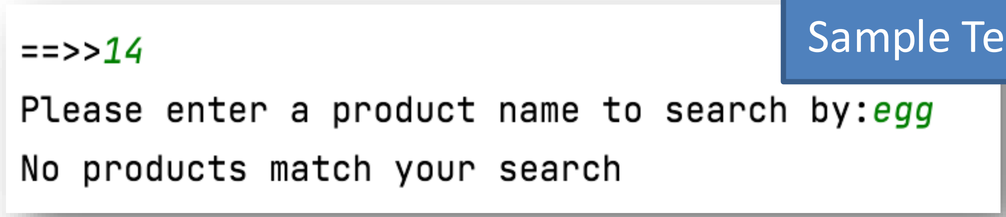
```
List of Products are:
0: Product description: 32 Inch TV, product code: 3204, unit cost: 199.44, currently in product line: N
1: Product description: Peanut Butter, product code: 3422, unit cost: 23.33, currently in product line: Y
2: Product description: Almond Butter, product code: 3432, unit cost: 45.33, currently in product line: Y
3: Product description: Walnut butter, product code: 4332, unit cost: 23.22, currently in product line: Y
4: Product description: iPhone, product code: 3323, unit cost: 54.33, currently in product line: N
```

Sample Test 1



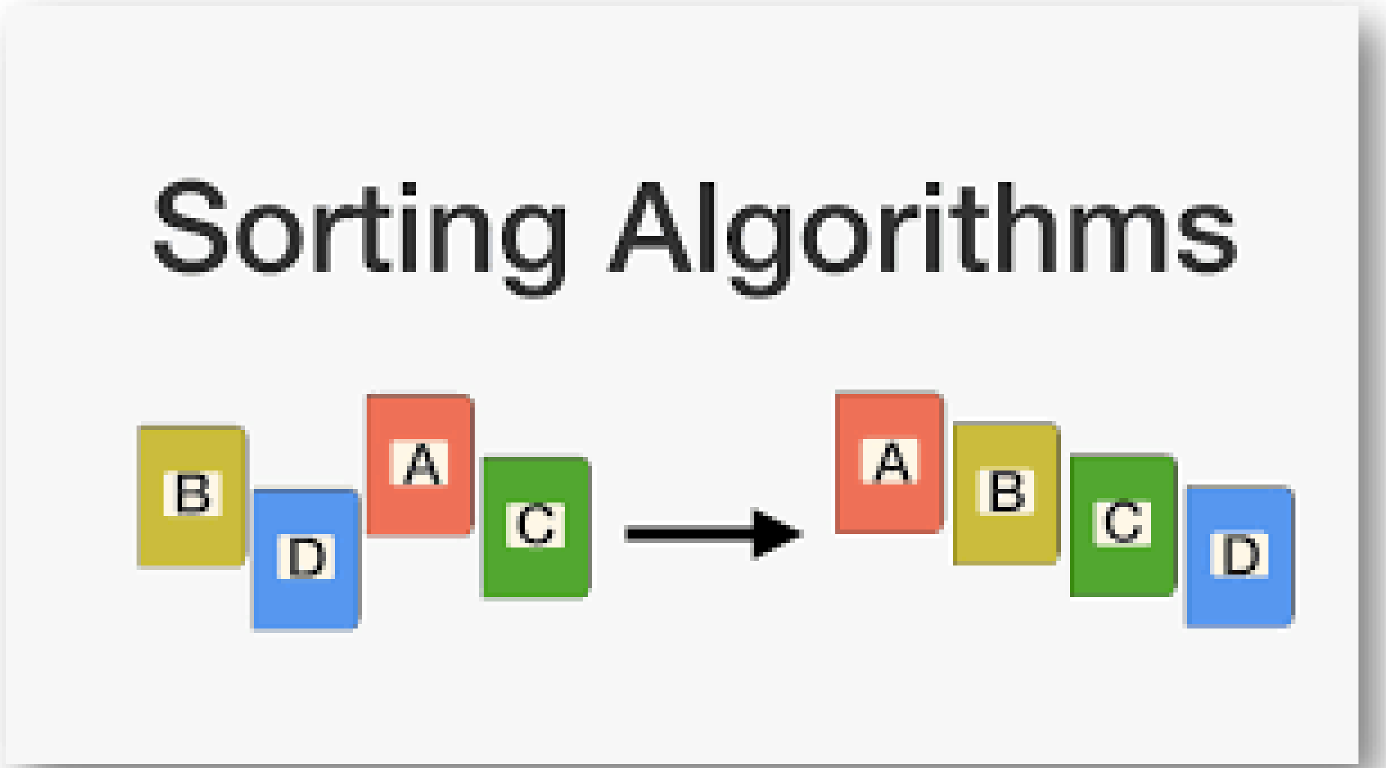
```
==>>14
Please enter a product name to search by:butter
1: Product description: Peanut Butter, product code: 3422, unit cost: 23.33, currently in product line: Y
2: Product description: Almond Butter, product code: 3432, unit cost: 45.33, currently in product line: Y
3: Product description: Walnut butter, product code: 4332, unit cost: 23.22, currently in product line: Y
```

Sample Test 2



```
==>>14
Please enter a product name to search by:egg
No products match your search
```

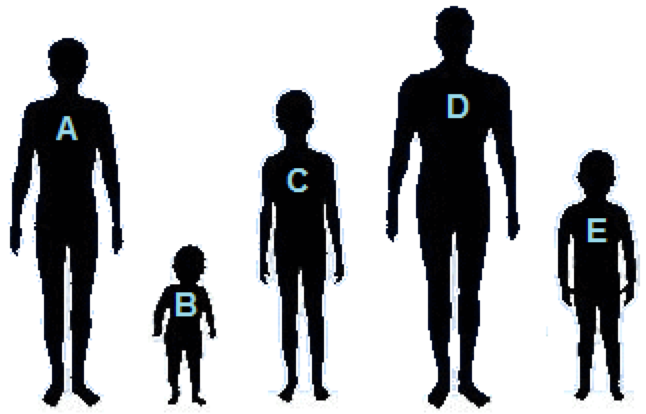
Problem: We want to search the **products** ArrayList and return all the individual products whose product name matches, or partially matches the search word entered by the user.



Selection Sort

Sorting

- The human brain can very quickly sort a collection e.g. it's straightforward to sort a group of people in height order.



- For a computer to carry out this process is more difficult.

Sorting

- A computer is quite limited in the data it can handle simultaneously and is restricted to:
  1. Comparing two items
  2. Swapping the items or copying one item.

- In the sorting algorithm we will investigate, we will see the above two steps repeatedly used.

ArrayList – Swapping Values

- Before we start discussing sorting, we will look at swapping two values in an ArrayList.

- This swap algorithm is needed during sorting.

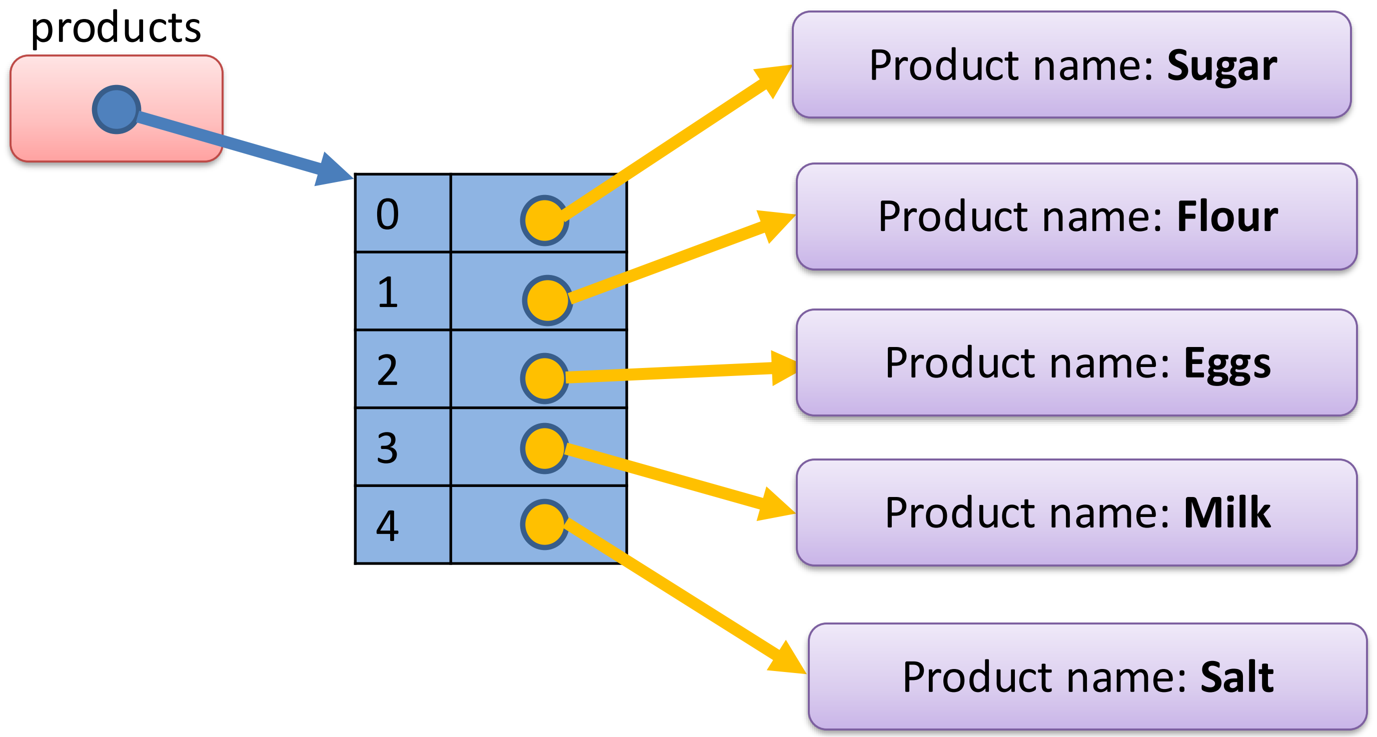
ArrayList - Swapping Values

```
private ArrayList<Product> products = new ArrayList<Product>();
```

ArrayList - Swapping Values

```
private ArrayList<Product> products = new ArrayList<Product>();
```

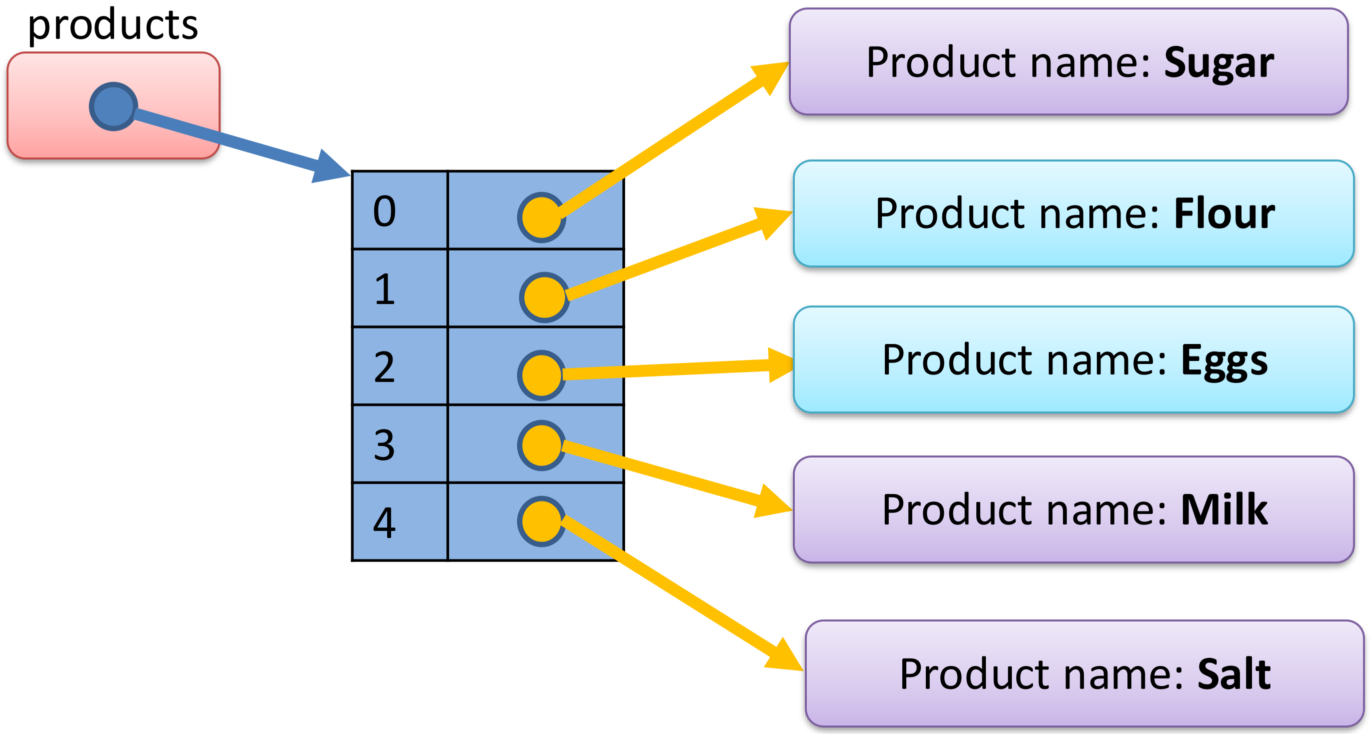
With five product objects added to the ArrayList:



ArrayList - Swapping Values

```
private ArrayList<Product> products = new ArrayList<Product>();
```

We now want to swap the objects Flour and Eggs with each other.



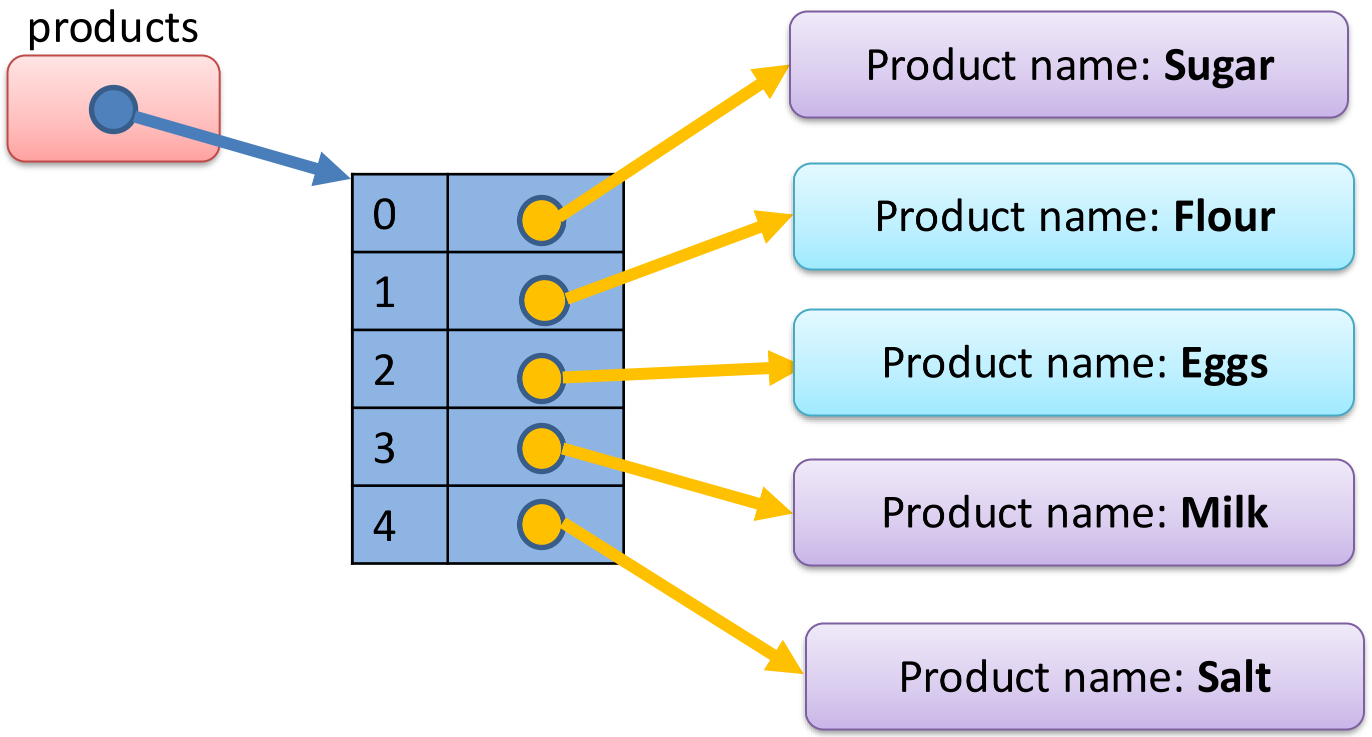
ArrayList - Swapping Values

```
private ArrayList<Product> products = new ArrayList<Product>();
```

Flour is index 1
Eggs is index 2

Algorithm:

a) **Store Flour and Eggs in temporary Product variables.**
b) **Set index 1 to be the temporary Product variable holding Eggs.**
c) **Set index 2 to be the temporary Product variable holding Flour.**



Algorithm

a) **Store Flour and Eggs in temporary Product variables.**
b) **Set index 1 to be the temporary Product variable holding Eggs.**
c) **Set index 2 to be the temporary Product variable holding Flour.**

```
private void swapProducts(ArrayList<Product> products,
                                int currentIndex, int highestIndex) {

    Product currentProduct = products.get(currentIndex);
    Product highestProduct = products.get(highestIndex);

    products.set(currentIndex, highestProduct);
    products.set(highestIndex,currentProduct);
}
```

a
b
c

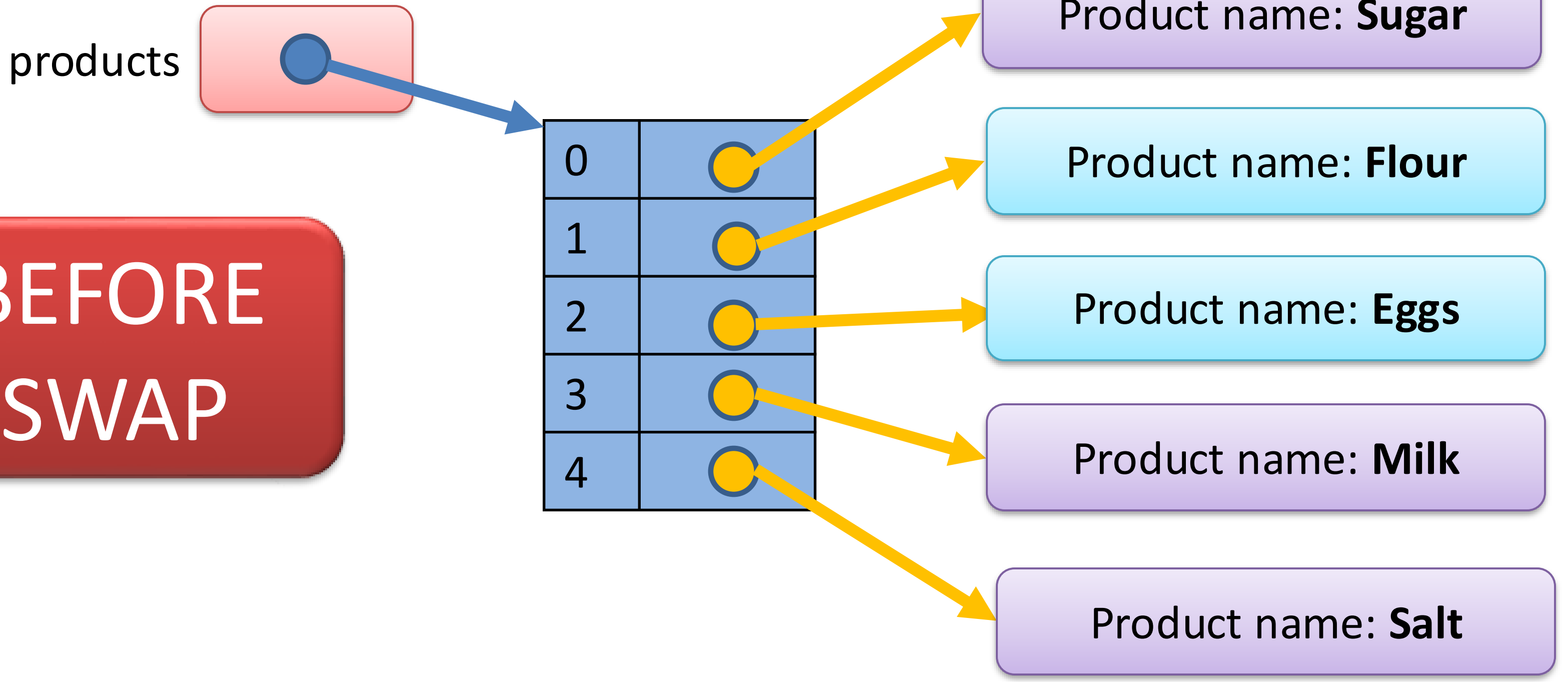
Algorithm

a) **Store Flour and Eggs in temporary Product variables.**
b) **Set index 1 to be the temporary Product variable holding Eggs.**
c) **Set index 2 to be the temporary Product variable holding Flour.**

```
private void swapProducts(ArrayList<Product> products,
                          int currentIndex, int highestIndex) {

a   Product currentProduct = products.get(currentIndex);
    Product highestProduct = products.get(highestIndex);

b   products.set(currentIndex, highestProduct);
c   products.set(highestIndex,currentProduct);
}
```



BEFORE SWAP

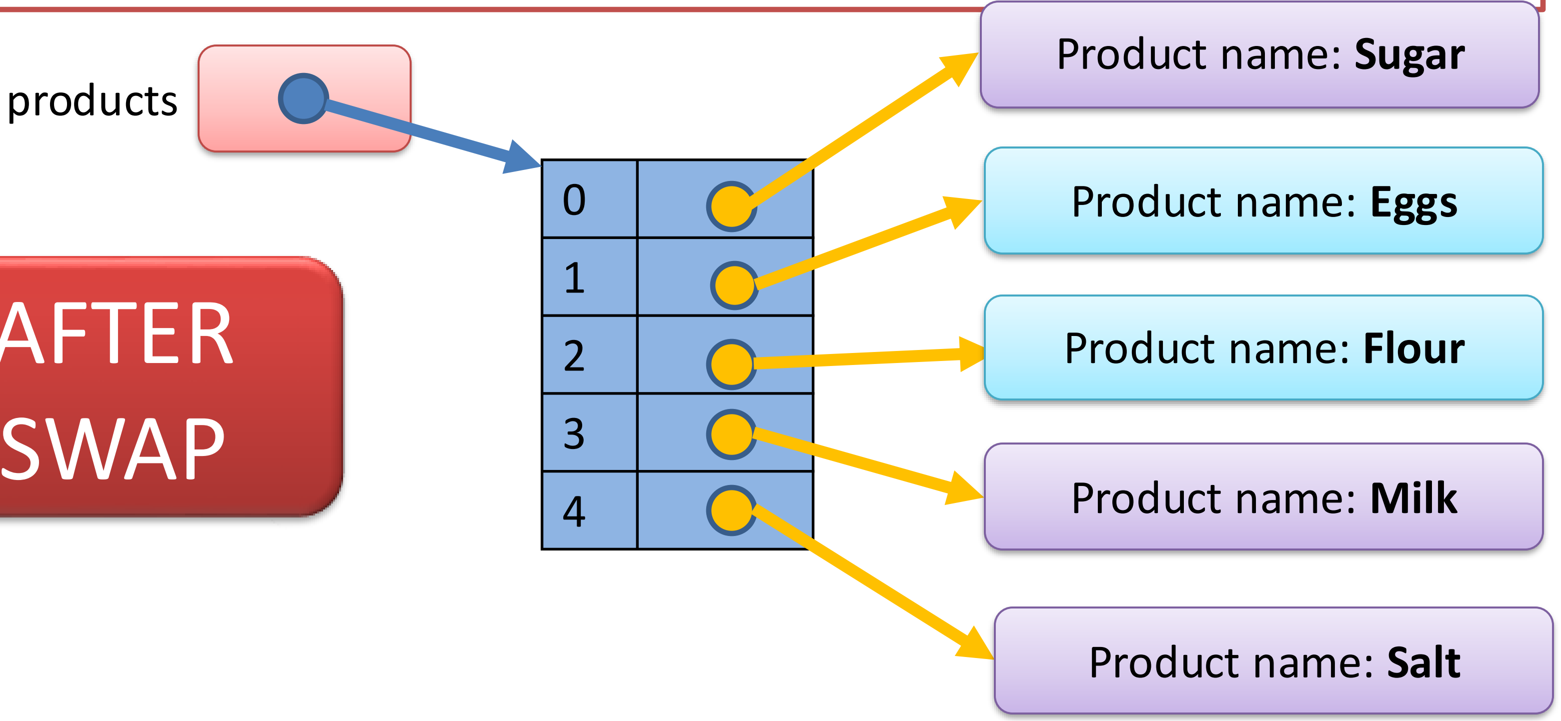
Algorithm

a) **Store Flour and Eggs in temporary Product variables.**
b) **Set index 1 to be the temporary Product variable holding Eggs.**
c) **Set index 2 to be the temporary Product variable holding Flour.**

```
private void swapProducts(ArrayList<Product> products,
                          int currentIndex, int highestIndex) {

a    Product currentProduct = products.get(currentIndex);
     Product highestProduct = products.get(highestIndex);

b    products.set(currentIndex, highestProduct);
c    products.set(highestIndex,currentProduct);
}
```



AFTER SWAP

Sorting

- Now that we know how to swap items in an ArrayList, we can start looking at sorting Algorithms.

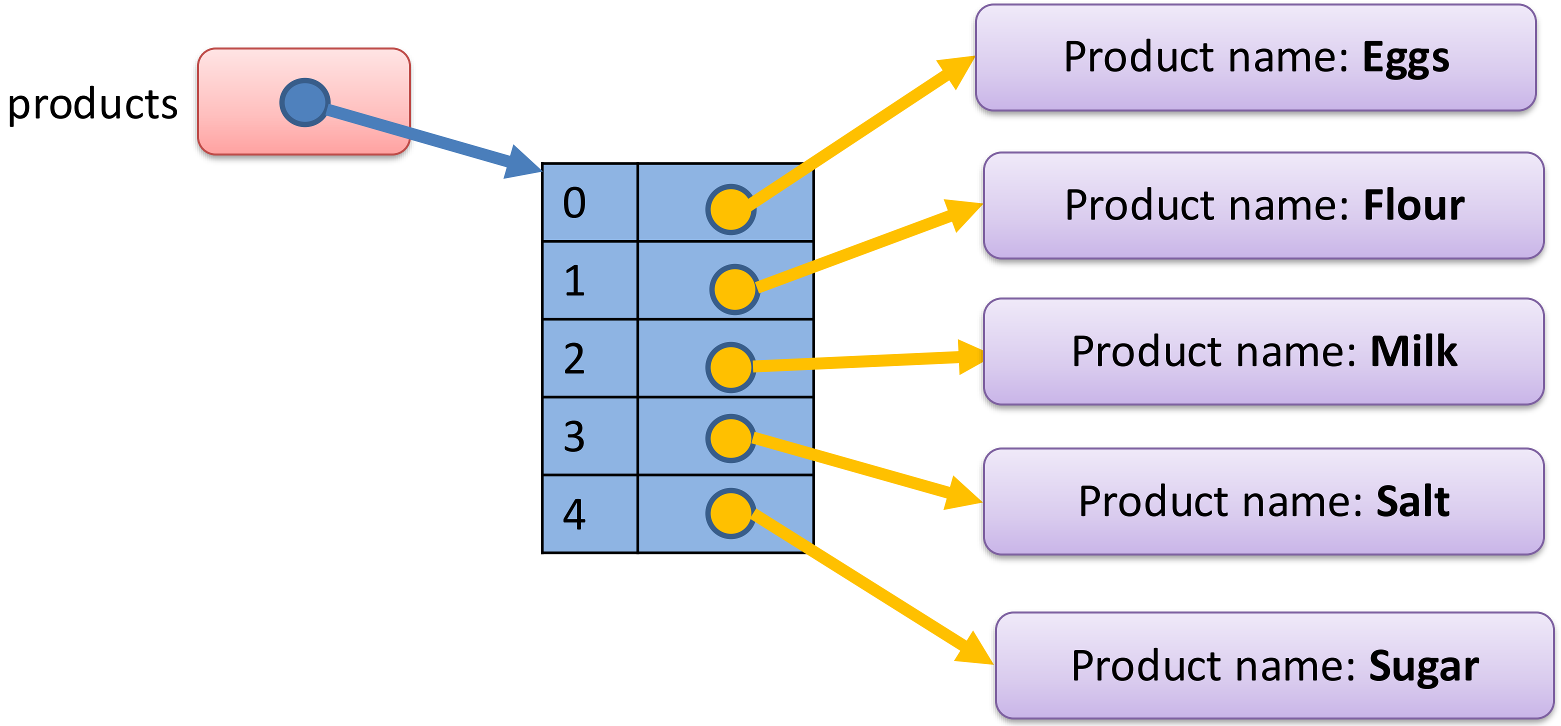
- Many basic sorting algorithms e.g.:
  1. Bubble sort
  2. Selection sort
  3. Insertion sort

-

What is a “Sorted List”?

- We can define a sorted list as a list in which each element is in its correct position, its correct position being defined as:

All values to the left of the element are less than or equal to it and all values to the right of the element are greater than or equal to it.



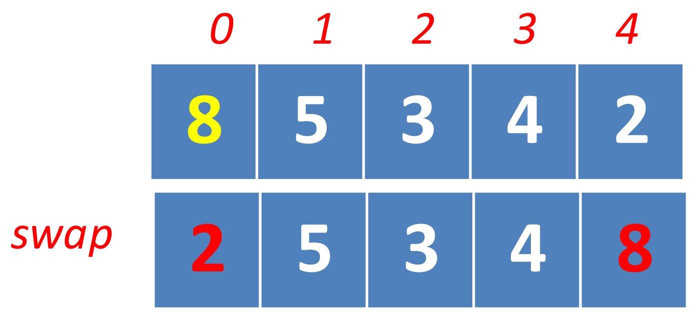
All values to the left of the element are less than or equal to it and all values to the right of the element are greater than or equal to it.

0	1	2	3	4
8	5	3	4	2

Selection Sort Algorithm

1. Iterate through all elements of the array and select the largest one (this is stored in a temporary variable).

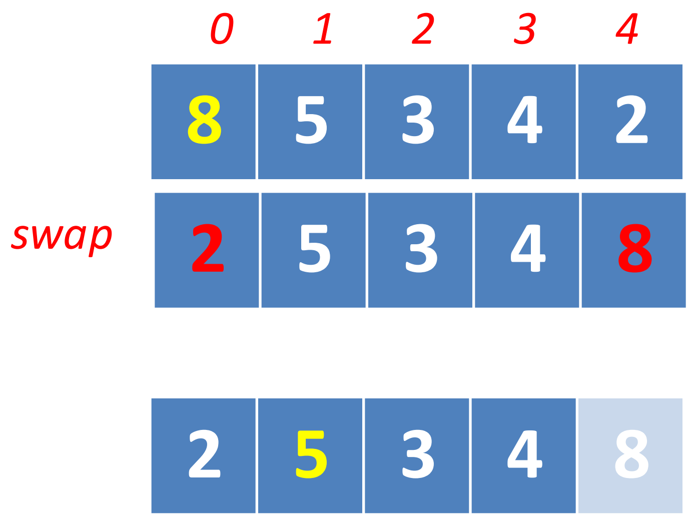
Selection Sort Algorithm



1. Iterate through all elements of the array and select the largest one (this is stored in a temporary variable).
2. The largest one is then swapped with the element at the end of the array.

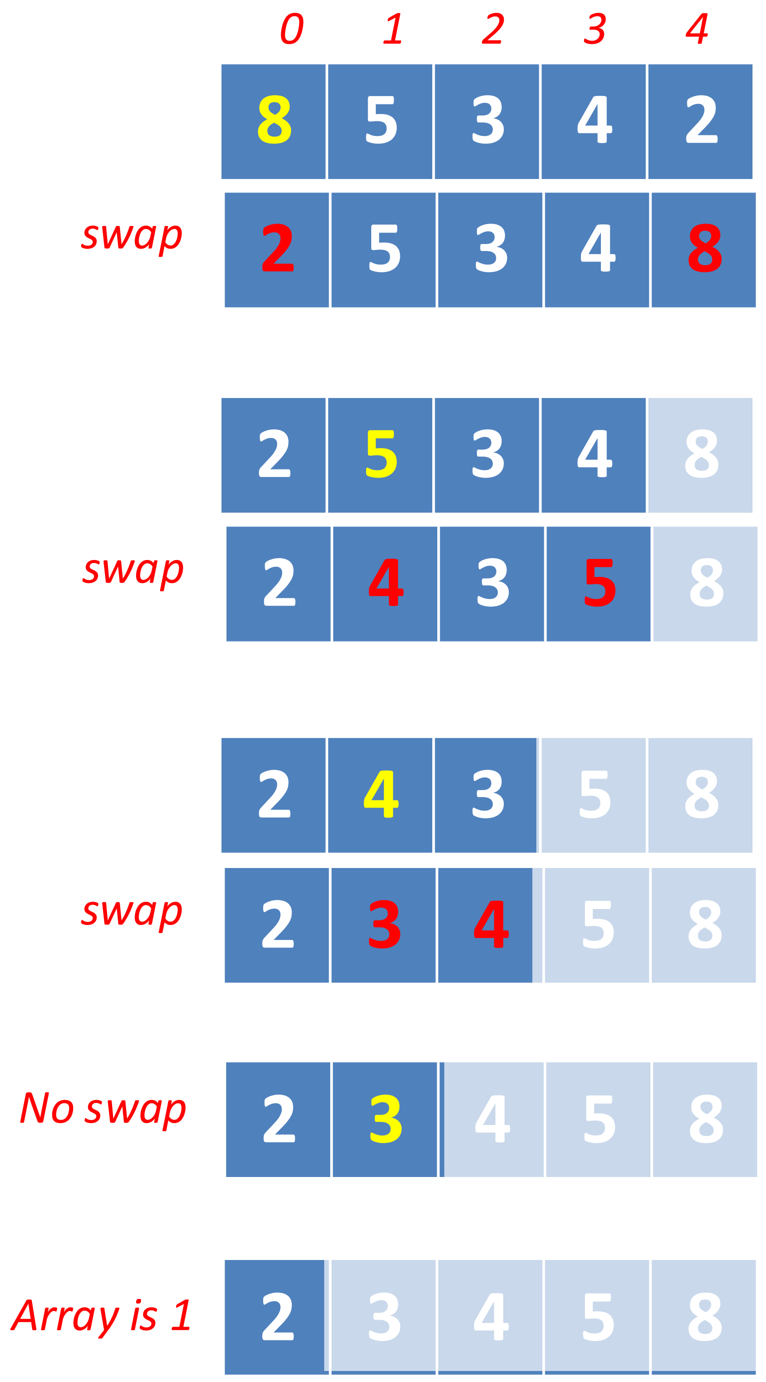
Selection Sort Algorithm

1. Iterate through all elements of the array and select the largest one (this is stored in a temporary variable).
2. The largest one is then swapped with the element at the end of the array.
3. Then reduce the size of the array by 1 and work through the array to locate the next largest element.



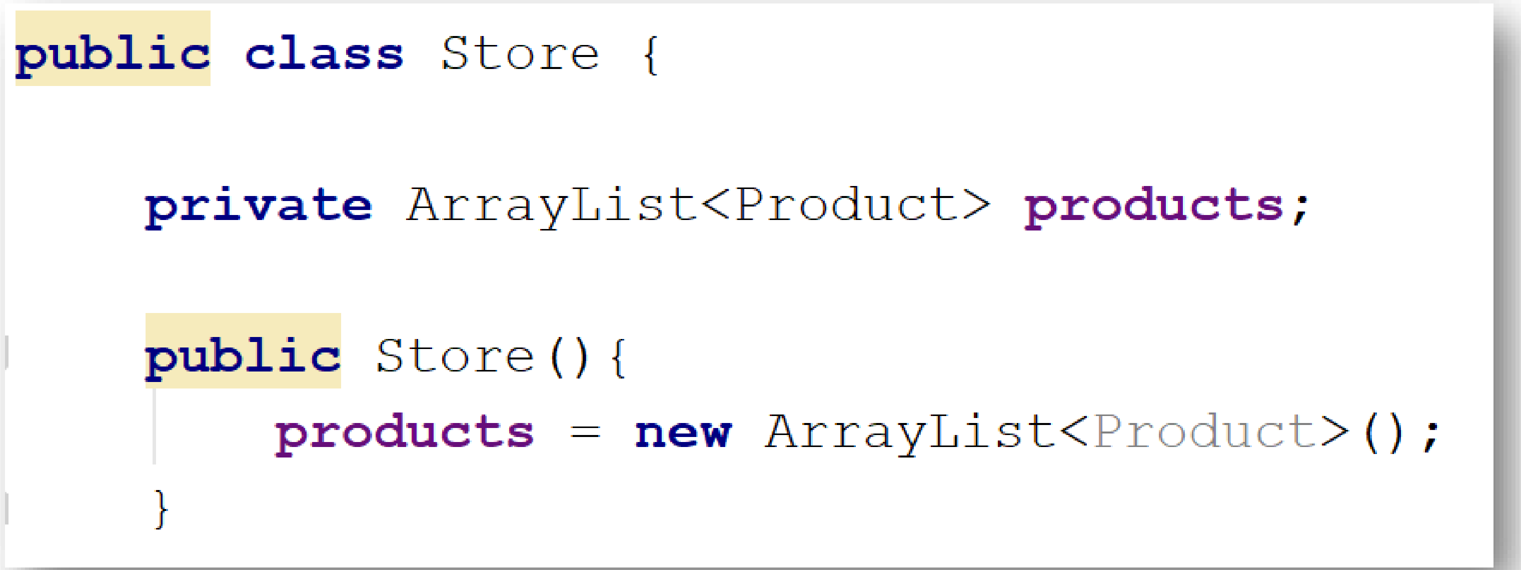
Selection Sort Algorithm

1. Iterate through all elements of the array and select the largest one (this is stored in a temporary variable).
2. The largest one is then swapped with the element at the end of the array.
3. Then reduce the size of the array by 1 and work through the array to locate the next largest element.
4. Repeat the above steps until the size of your array is 1.



Store.java

ShopV8.0 – Selection Search



```
public class Store {

    private ArrayList<Product> products;

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

Problem: We want to physically sort the **products** ArrayList by product name ascending. This will involve swapping in and out items in the ArrayList until we have a fully sorted collection.

Store.java

ShopV8.0 – Selection Search

```
public void sortProductsByNameAscending(){

    for (int i = products.size() -1; i >= 0; i--) {
        int highestIndex = 0;
        for (int j = 0; j <= i; j++) {
            if (products.get(j).getProductName().compareTo(
             products.get(highestIndex).getProductName()) > 0) {
                highestIndex = j;
            }
        }
        swapProducts(products, i, highestIndex);
    }
}
```

Problem: We want to physically sort the **products** ArrayList by product name ascending. This will involve swapping in and out items in the ArrayList until we have a fully sorted collection.

Store.java

ShopV8.0 – Selection Search

```
private void swapProducts(ArrayList<Product> products, int current, int highest) {
    Product smaller = products.get(current);
    Product bigger = products.get(highest);

    products.set(current, bigger);
    products.set(highest,smaller);
}
```

Problem: We want to physically sort the **products** ArrayList by product name ascending. This will involve swapping in and out items in the ArrayList until we have a fully sorted collection.

Driver.java

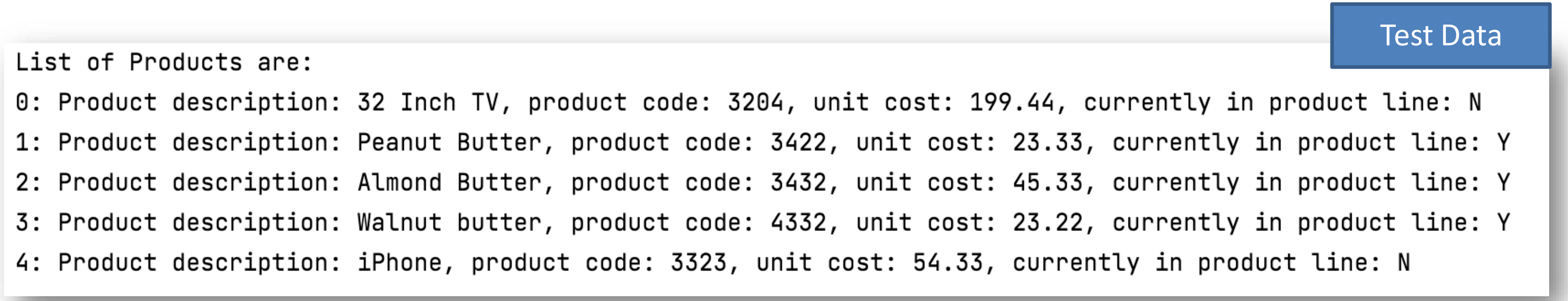
```java
private int mainMenu() {
    int option = ScannerInput.readNextInt("""
            ------------------------------------------------------------------
            |                          Shop Menu                             |
            ------------------------------------------------------------------
            | PRODUCT MENU                                                   |
            |   1) Add a product                                             |
            |   2) List the Products                                         |
            |   3) Update a product                                          |
            |   4) Delete a product                                          |
            ------------------------------------------------------------------
            | PRODUCT DESCRIPTION MENU                                       |
            |   5) Add a product description                                 |
            |   6) List product description(s)                               |
            |   7) Update a product description                              |
            |   8) Delete a product description                              |
            ------------------------------------------------------------------
            | REPORT MENU                                                    |
            |   10) List the current products                                |
            |   11) Display average product unit cost                        |
            |   12) Display cheapest product                                 |
            |   13) List products that are more expensive than a given price |
            ------------------------------------------------------------------
            | SEARCH AND SORT MENU                                           |
            |   14) Search products by name                                  |
            |   15) Sort products by name ascending                          |
            ------------------------------------------------------------------
            |   20) Save products to products.xml                            |
            |   21) Load products from products.xml                          |
            |   0)  Exit                                                     |
            ------------------------------------------------------------------
            ==>>  """);
    return option;
}
```

```java
switch (option) {
    case 1 -> addProduct();
    case 2 -> printProducts();
    case 3 -> updateProduct();
    case 4 -> deleteProduct();
    case 5 -> addDescriptionToProduct();
    case 6 -> listProductDescriptions();
    case 7 -> updateDescriptionInProduct();
    case 8 -> deleteDescriptionFromProduct();
    case 10 -> printCurrentProducts();
    case 11 -> printAverageProductPrice();
    case 12 -> printCheapestProduct();
    case 13 -> printProductsAboveAPrice();
    case 14 -> searchProductByName();
    case 15 -> sortProductByName();
    case 20 -> saveProducts();
    case 21 -> loadProducts();
    default -> System.out.println("Invalid option entered: " + option);
}
```

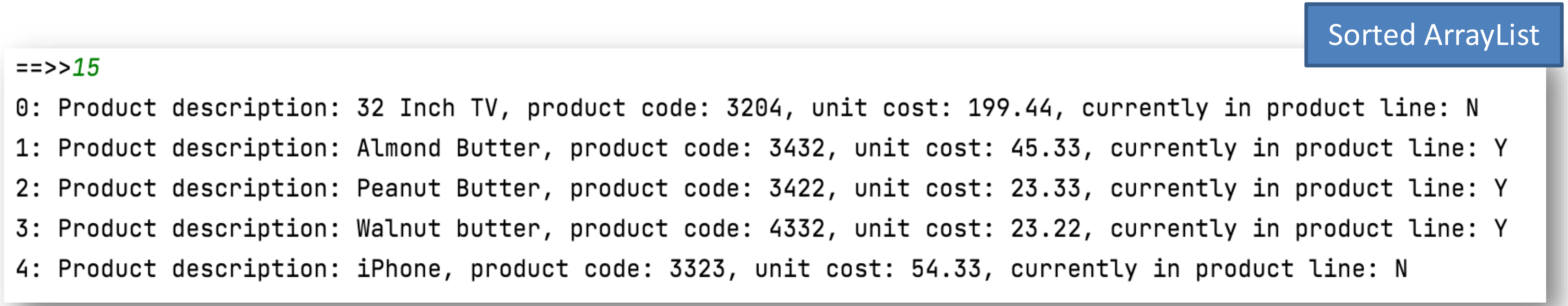
Driver.java

```
private void sortProductByName() {
    store.sortProductsByNameAscending();
    System.out.println(store.listProducts());
}
```

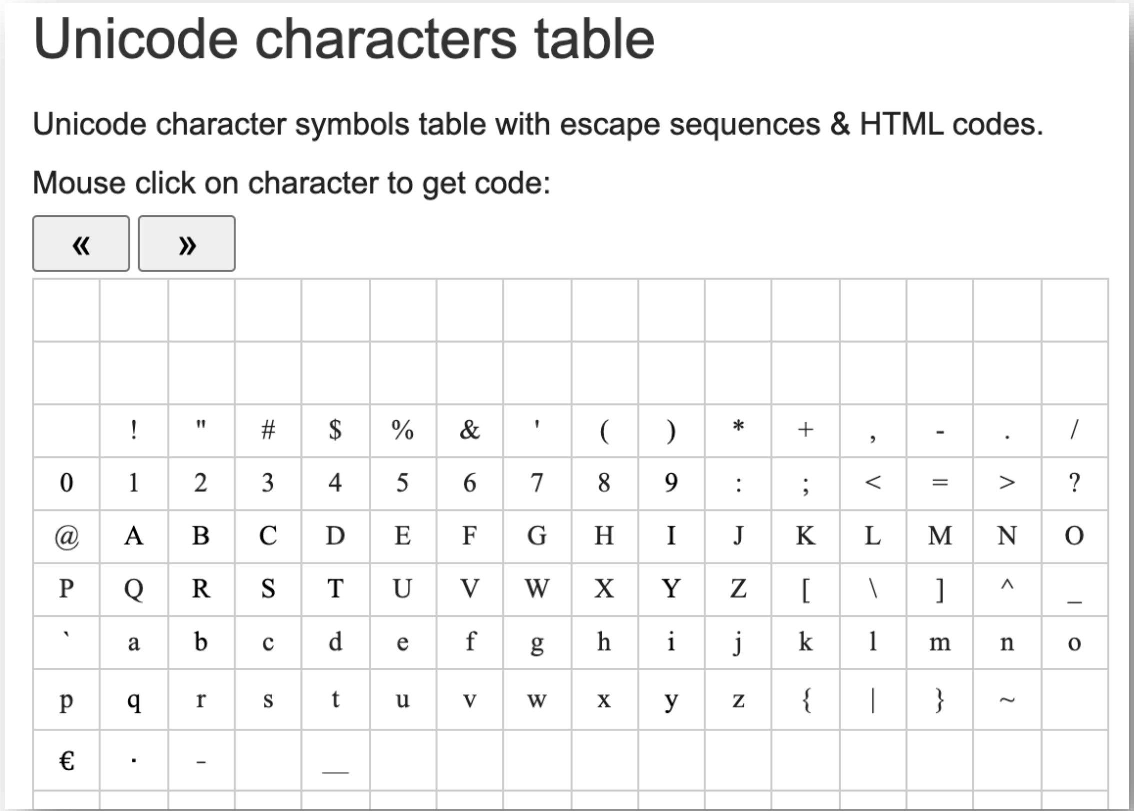
```
switch (option) {
    case 1 -> addProduct();
    case 2 -> printProducts();
    case 3 -> updateProduct();
    case 4 -> deleteProduct();
    case 5 -> addDescriptionToProduct();
    case 6 -> listProductDescriptions();
    case 7 -> updateDescriptionInProduct();
    case 8 -> deleteDescriptionFromProduct();
    case 10 -> printCurrentProducts();
    case 11 -> printAverageProductPrice();
    case 12 -> printCheapestProduct();
    case 13 -> printProductsAboveAPrice();
    case 14 -> searchProductByName();
    case 15 -> sortProductByName();
    case 20 -> saveProducts();
    case 21 -> loadProducts();
    default -> System.out.println("Invalid option entered: " + option);
}
```


Test Data

```
List of Products are:
0: Product description: 32 Inch TV, product code: 3204, unit cost: 199.44, currently in product line: N
1: Product description: Peanut Butter, product code: 3422, unit cost: 23.33, currently in product line: Y
2: Product description: Almond Butter, product code: 3432, unit cost: 45.33, currently in product line: Y
3: Product description: Walnut butter, product code: 4332, unit cost: 23.22, currently in product line: Y
4: Product description: iPhone, product code: 3323, unit cost: 54.33, currently in product line: N
```


Sorted ArrayList

```
==>>15
0: Product description: 32 Inch TV, product code: 3204, unit cost: 199.44, currently in product line: N
1: Product description: Almond Butter, product code: 3432, unit cost: 45.33, currently in product line: Y
2: Product description: Peanut Butter, product code: 3422, unit cost: 23.33, currently in product line: Y
3: Product description: Walnut butter, product code: 4332, unit cost: 23.22, currently in product line: Y
4: Product description: iPhone, product code: 3323, unit cost: 54.33, currently in product line: N
```



Unicode characters table

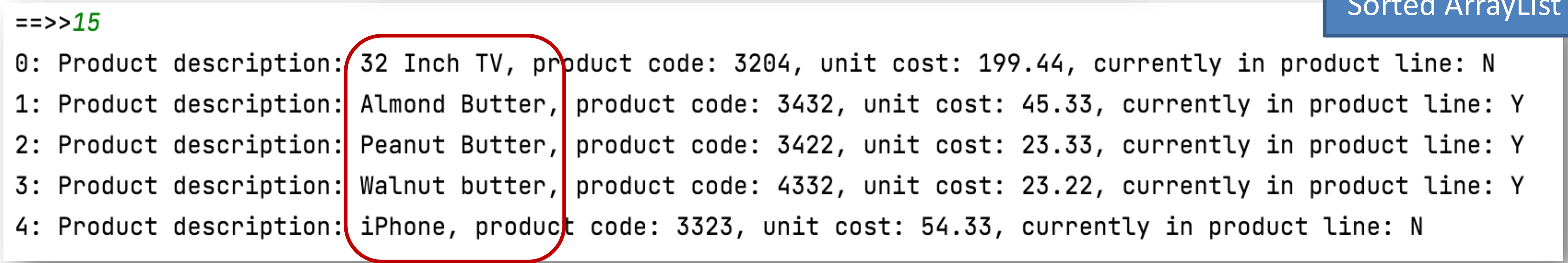
Unicode character symbols table with escape sequences & HTML codes.

Mouse click on character to get code:

« »

	!	"	#	$	%	&	'	(	)	*	+	,	-	.	/
0	1	2	3	4	5	6	7	8	9	:	;	<	=	>	?
@	A	B	C	D	E	F	G	H	I	J	K	L	M	N	O
P	Q	R	S	T	U	V	W	X	Y	Z	[	\	]	^	_
`	a	b	c	d	e	f	g	h	i	j	k	l	m	n	o
p	q	r	s	t	u	v	w	x	y	z	{	|	}	~	
€	·	-		__											

Sorted ArrayList



```
==>>15
0: Product description: 32 Inch TV, product code: 3204, unit cost: 199.44, currently in product line: N
1: Product description: Almond Butter, product code: 3432, unit cost: 45.33, currently in product line: Y
2: Product description: Peanut Butter, product code: 3422, unit cost: 23.33, currently in product line: Y
3: Product description: Walnut butter, product code: 4332, unit cost: 23.22, currently in product line: Y
4: Product description: iPhone, product code: 3323, unit cost: 54.33, currently in product line: N
```

THANK YOU!