1

Deploying and Maintaining the Duke's Choice Application

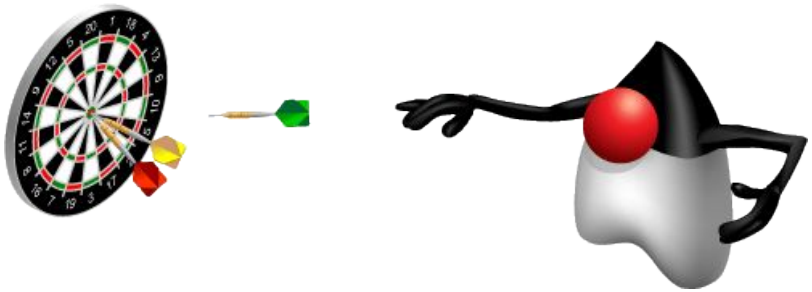
4

ORACLE

Objectives

After completing this lesson, you should be able to do the following:

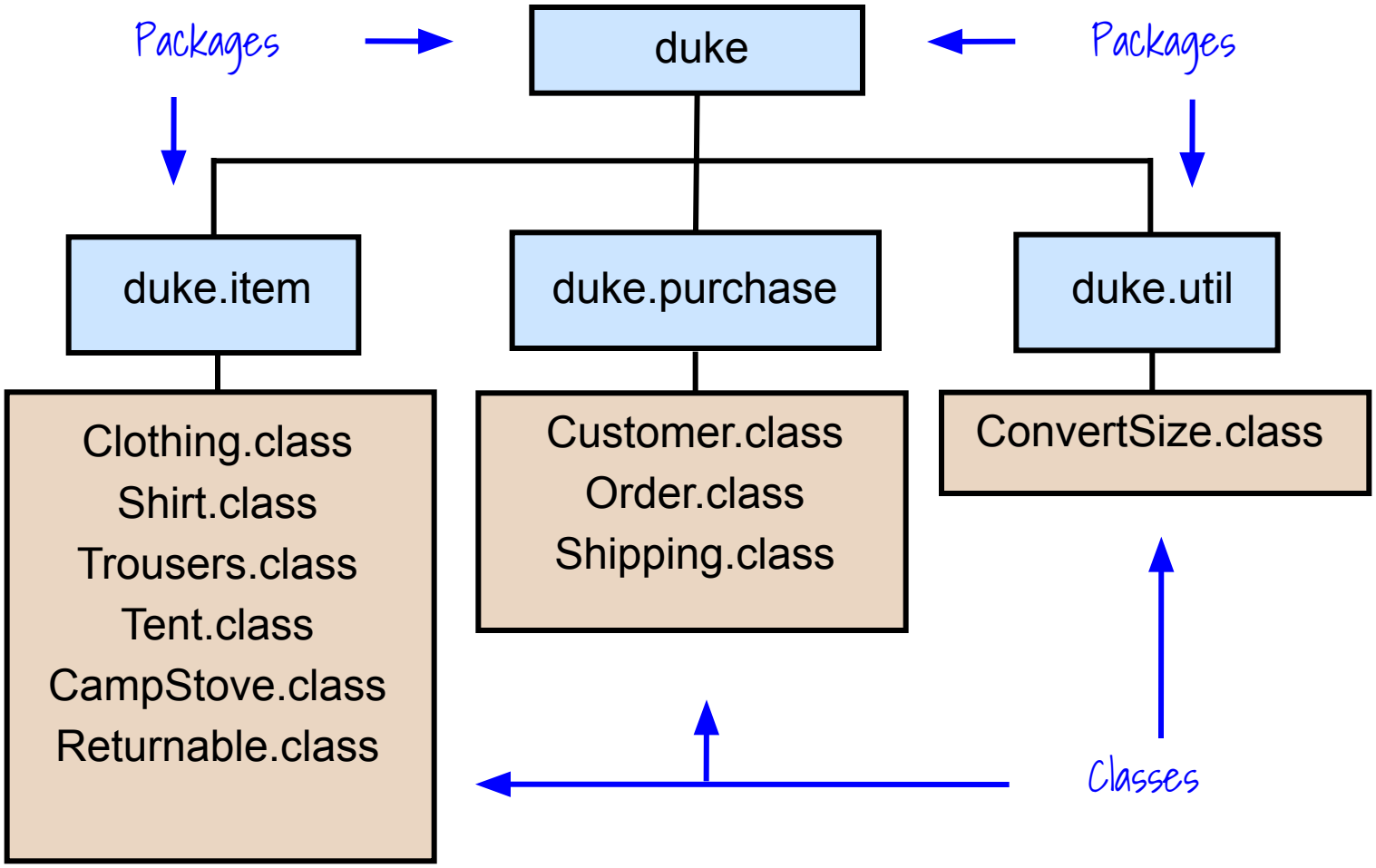
- Deploy a simple application as a JAR file
- Describe the parts of a Java application, including the user interface and the back end
- Describe how classes can be extended to implement new capabilities in the application



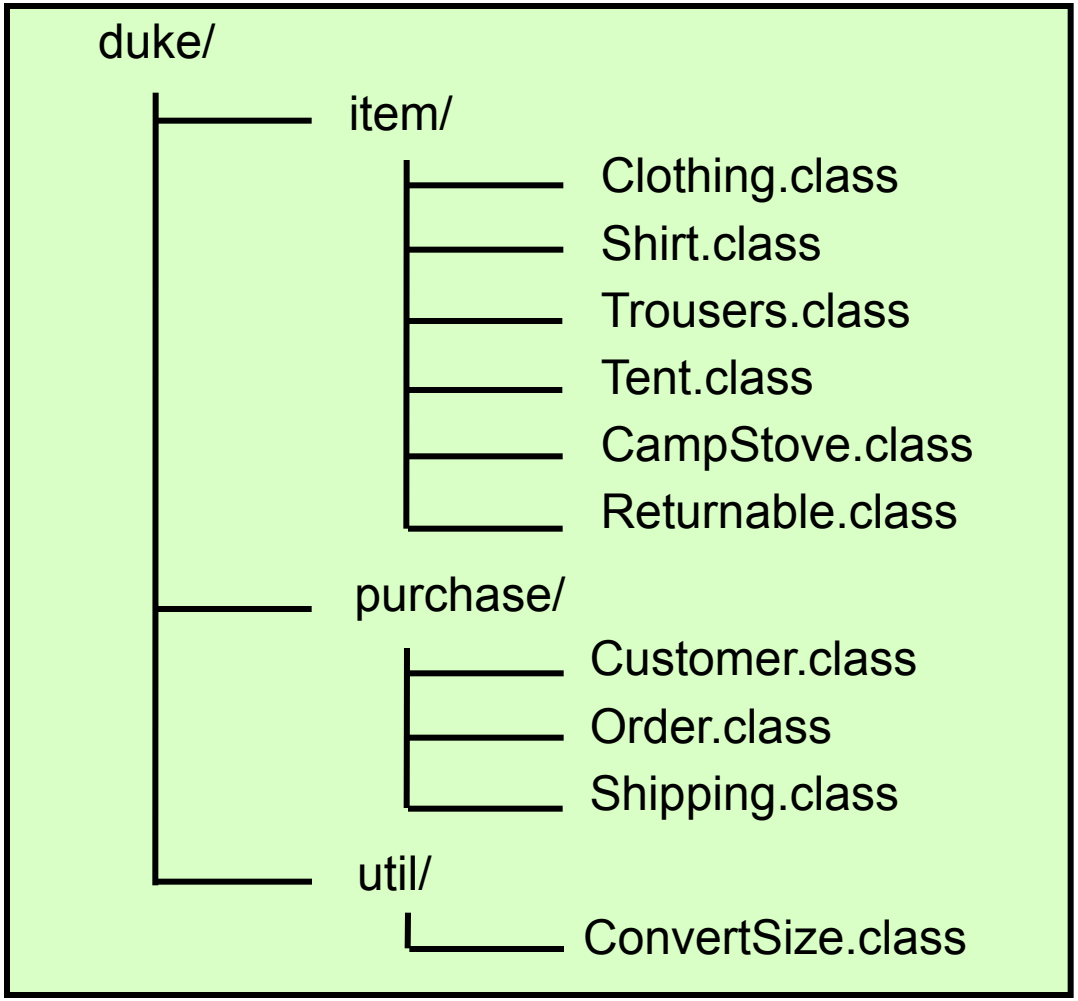
Topics

- Packages
- JARs and deployment
- Two-tier and three-tier architecture
- The Duke's Choice application
- Application modifications and enhancements

Packages



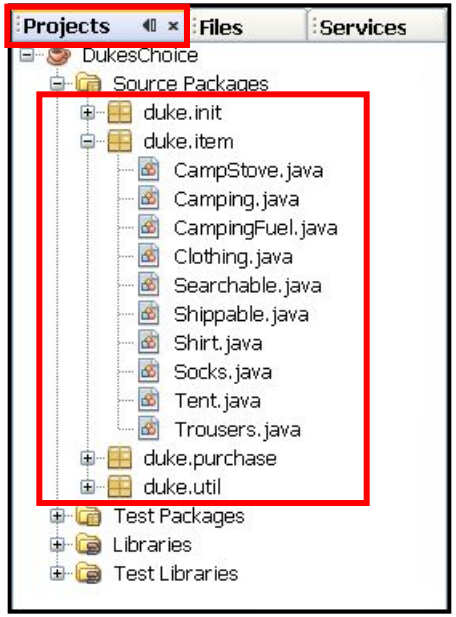
Packages Directory Structure



Packages in NetBeans

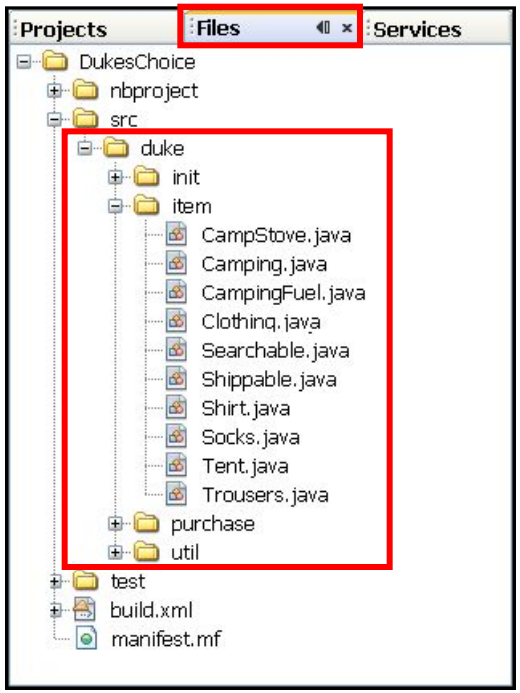
Projects tab

Files tab



Packages shown as icons

File structure for packages shown



Packages in Source Code

This class is in the package duke.item.

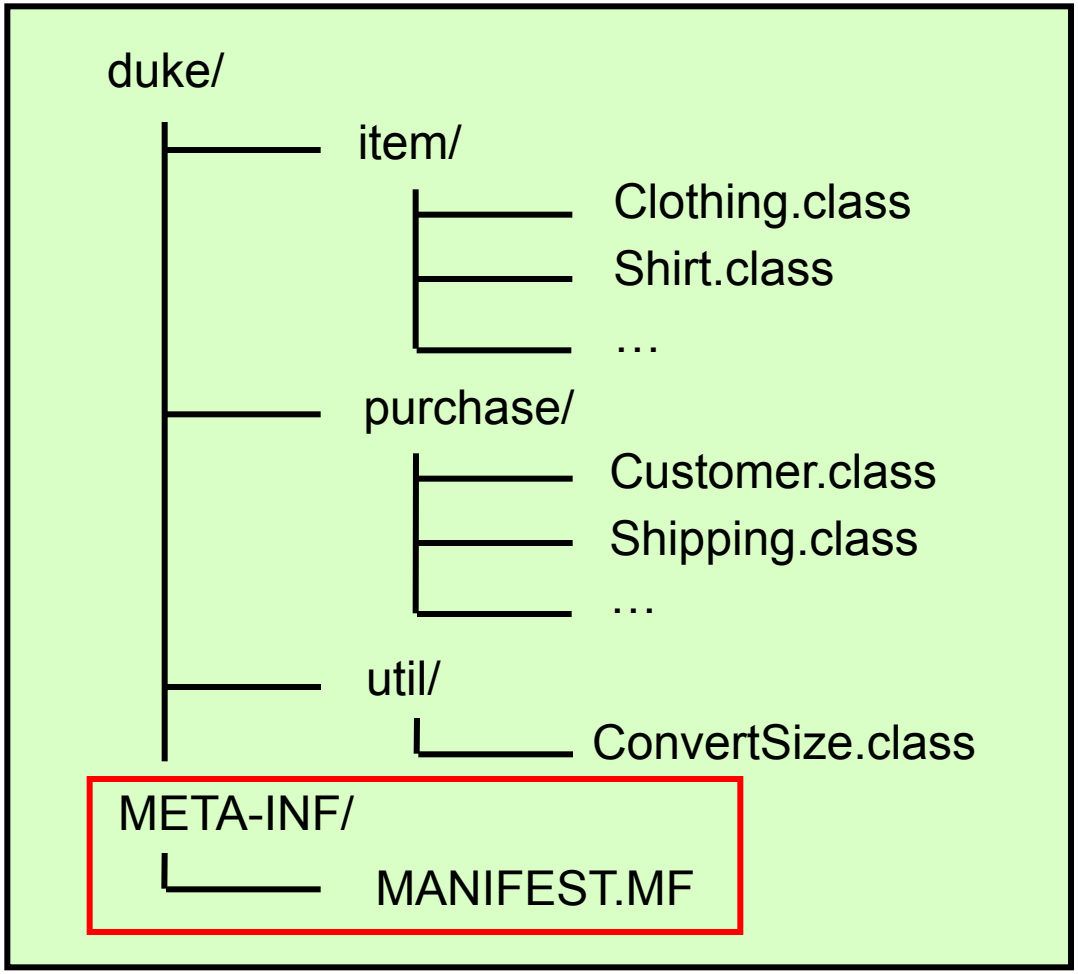
```
package duke.item;

public abstract class Clothing  implements Searchable, Shippable {
    private int itemID = 0;
    private String description = "-description required-";
    private char colorCode = 'U';

    ... < remaining code omitted > ...
```

The package that a class belongs to is defined in the source code.

Topics

- Packages
- JARs and deployment
- Two-tier and three-tier architecture
- The Duke's Choice application
- Application modifications and enhancements

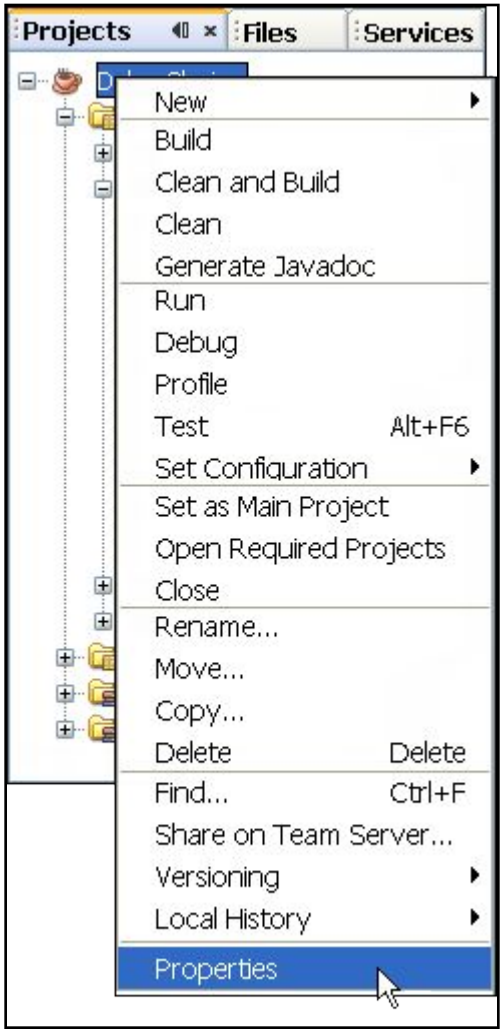
DukesChoice.jar

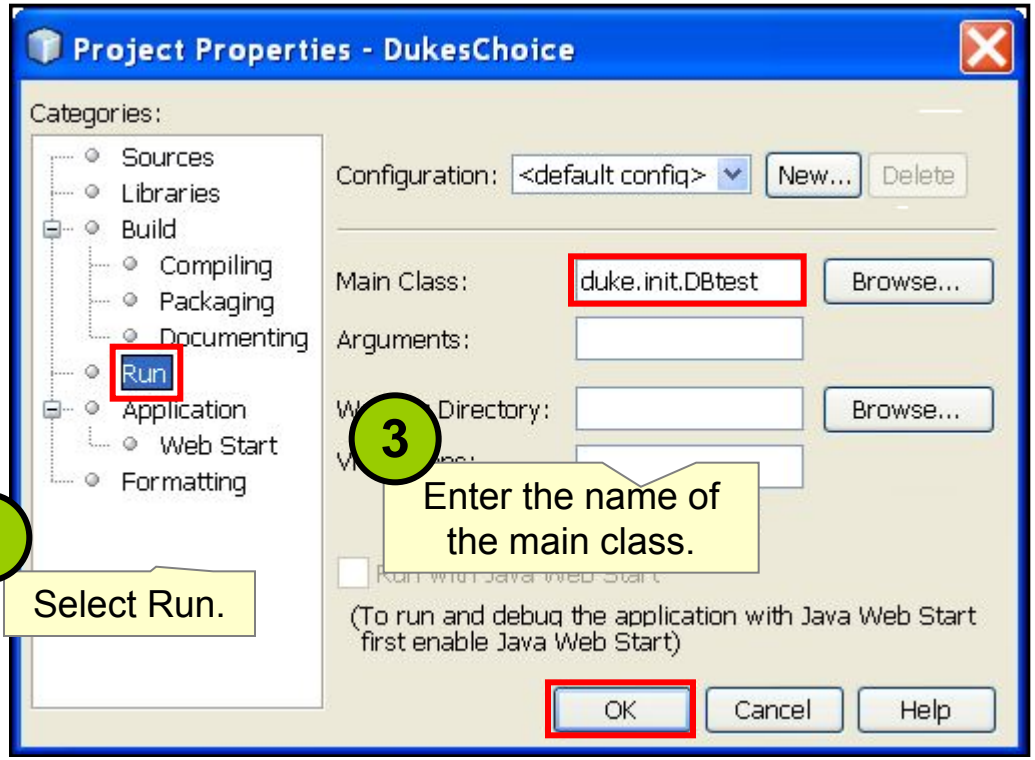


The JAR file contains
the class directory structure plus
a manifest file.

Manifest file
MANIFEST.MF
added

Set Main Class of Project





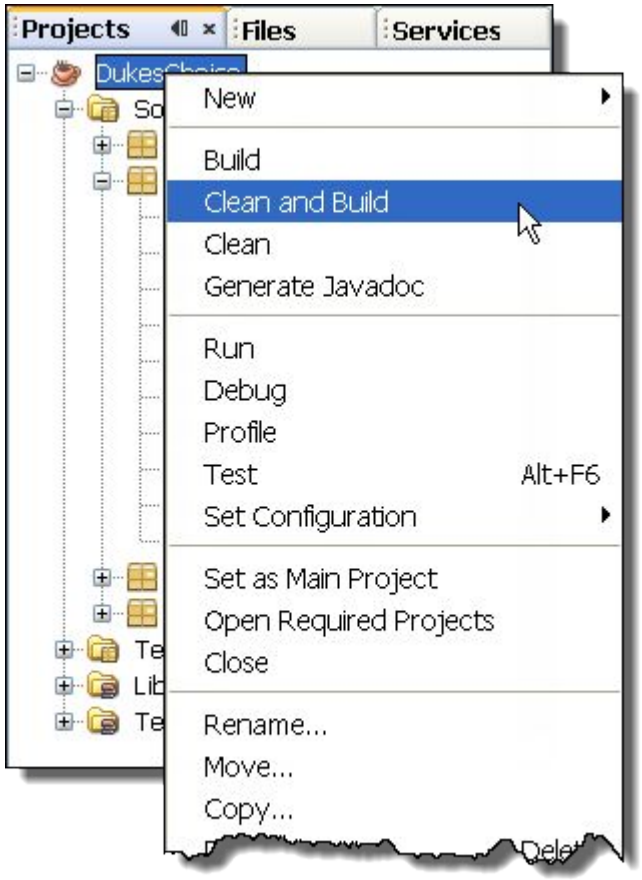
2 Select Run.

3 Enter the name of the main class.

1 Right-click the project and select Properties.

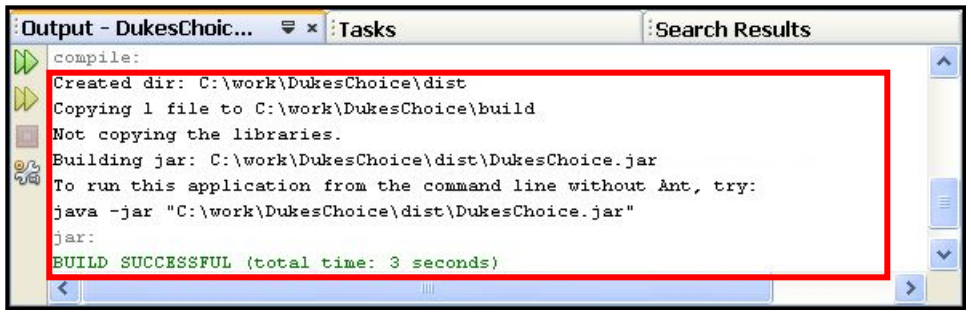
4 Click OK.

Creating the JAR File with NetBeans



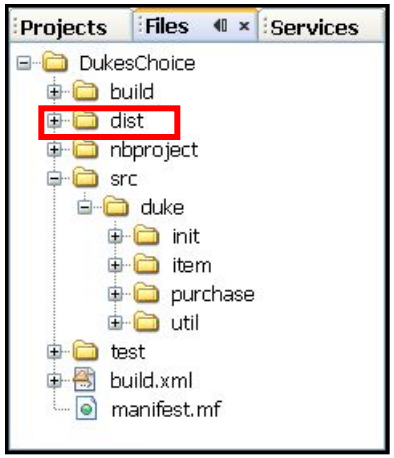
1 Right-click the project and select "Clean and Build."

2 Check the output to ensure the build is successful.

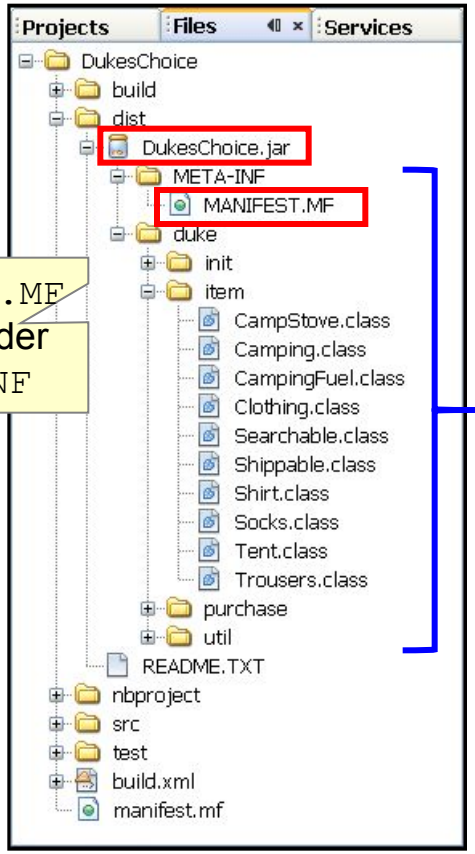


Creating the JAR File with NetBeans

Now a new directory in the Project



`MANIFEST.MF` added under `META-INF`



`DukesChoice.jar` under `dist` directory

The JAR file contains the class directory structure plus a manifest file.

Topics

- Packages
- JARs and deployment
- Two-tier and three-tier architecture
- The Duke's Choice application
- Application modifications and enhancements

Client/Server Two-Tier Architecture

Client/server computing involves two or more computers sharing tasks:

- Each computer performs logic appropriate to its design and stated function.
- The front-end client communicates with the back-end database.
- Client requests data from back end.
- Server returns appropriate results.
- Client handles and displays data.

Client/Server Three-Tier Architecture

- Three-tier client/server is a more complex, flexible approach.
- Each tier can be replaced by a different implementation:
  - Presentation can be GUI, web, smartphone, or even console.
  - Business logic defines business rules.
  - Data tier is an encapsulation of all existing data sources.

Topics

- Packages
- JARs and deployment
- Two-tier and three-tier architecture
- The Duke's Choice application
- Application modifications and enhancements

The Duke's Choice Application

- Abstract classes
  - Clothing
    - Extended by Shirt and other clothing classes
  - Camping
    - Extended by Tent and other camping classes
- Interfaces
  - Searchable
    - All purchasable items implement Searchable.
  - Returnable
    - Items that can be returned implement Returnable.
  - Shippable
    - Items that can be shipped implement Shippable.

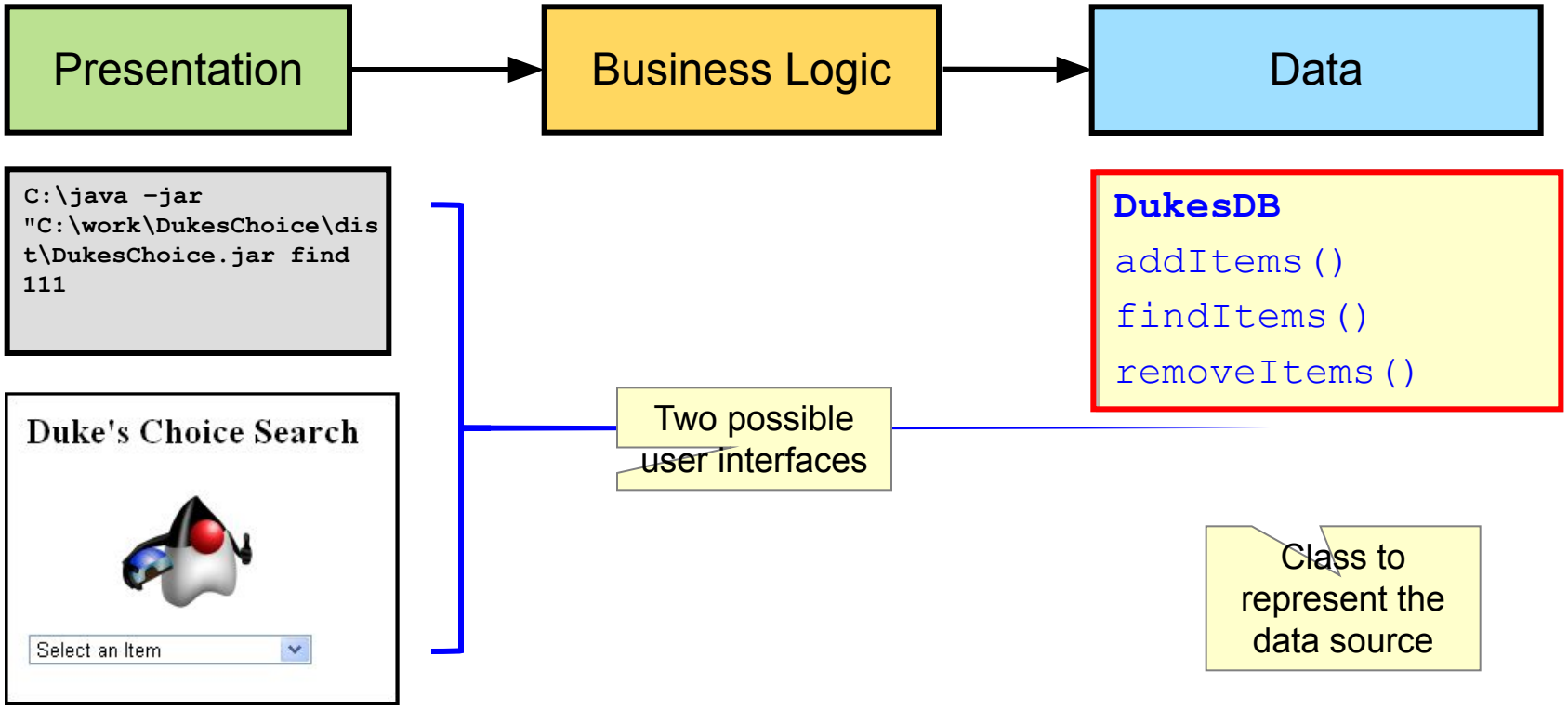
Clothing Class

```
package duke.item;
public abstract class Clothing  implements Searchable, Shippable {
    private String sku = "";
    private int itemID = 0; // Default ID for all clothing items
    private String description = "-description required-"; // default
    private char colorCode = 'U'; // Exception if invalid color code?
    private double price = 0.0; // Default price for all items
    private int quantityInStock = 0;

  public Clothing(int itemID, String description, char colorCode,
                  double price, int quantityInStock ) {
    this.itemID = itemID;
    this.description = description;
    this.colorCode = colorCode;
    this.price = price;
    this.quantityInStock = quantityInStock;
    this.sku = "" + itemID + colorCode;
        ... < more code follows > ...
```

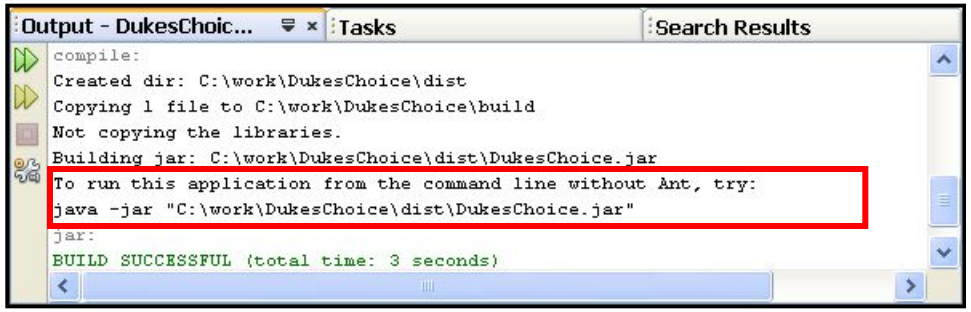
Clothing Class

```
public String getDisplay(String separator) {

  String displayString = "SKU: " + getSku() + separator +
  "Item: " + description + separator +
  "Price: " + price + separator +
  "Color: " + colorCode + separator +
  "Available: " + quantityInStock;
  return displayString;
}


      ... < more code follows > ...
```

Tiers of Duke's Choice



Running the JAR File from the Command Line



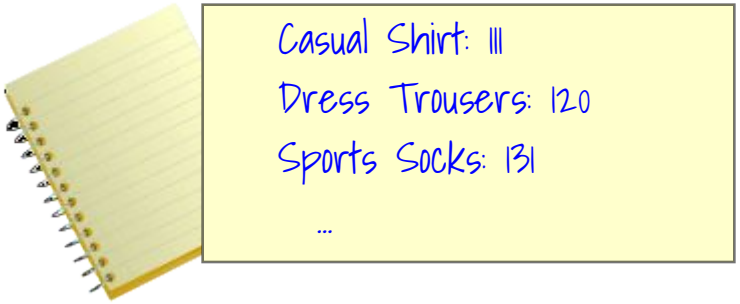
The command to run the JAR file

```
C:\java -jar "C:\work\DukesChoice\dist\DukesChoice.jar
```

Output:

```
Please add parameters in the format:
   find <item id number>
   OR
   remove <sku> <number to remove>
```

Listing Items from the Command Line

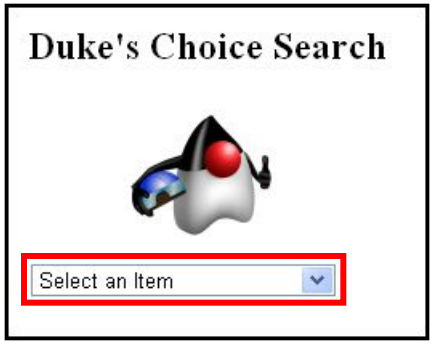


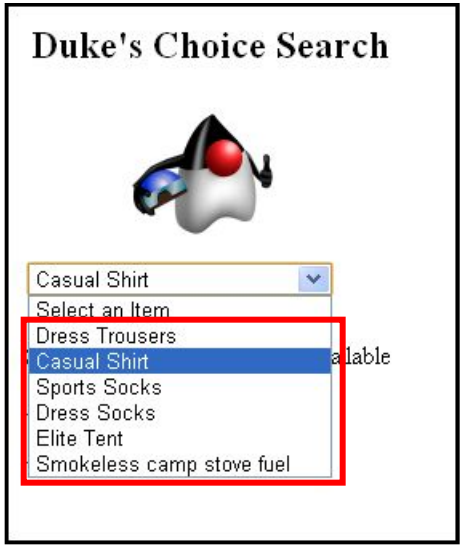

```
C:\java -jar "C:\work\DukesChoice\dist\DukesChoice.jar find 111
```

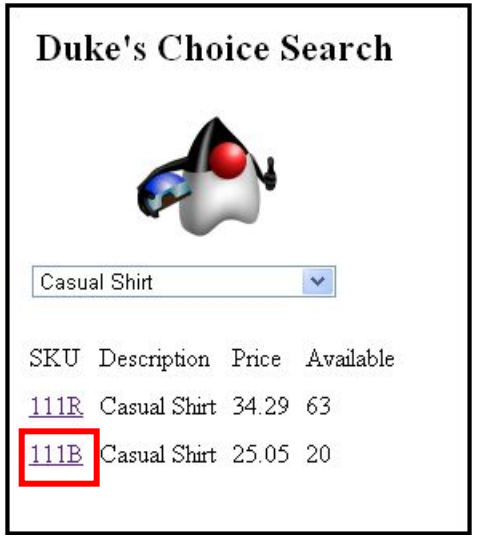
Output:

```
-----------------------------------------------------------------------
SKU: 111R | Item: Casual Shirt | Price: 34.29 | Color: R | Available: 63
-----------------------------------------------------------------------
SKU: 111B | Item: Casual Shirt | Price: 25.05 | Color: B | Available: 20
-----------------------------------------------------------------------
```

Listing Items in Duke's Choice Web Application







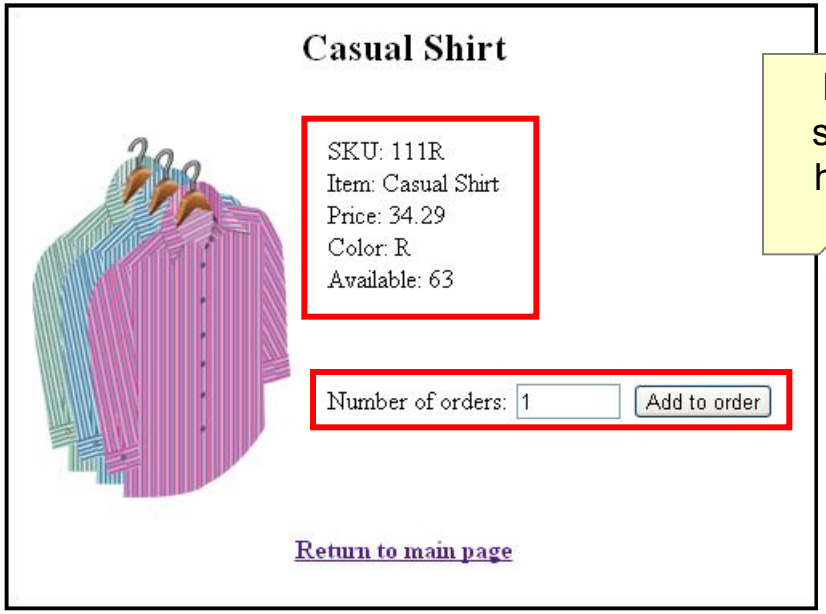
The Search page has a drop-down menu.

The current items in stock are shown.

Selecting an item displays a list of all those items.

The SKU for the item is an anchor tag.

Listing Items in Duke's Choice Web Application



Details of the shirt, including how many are available

Click to order

Topics

- Packages
- JARs and deployment
- Two-tier and three-tier architecture
- Duke's Choice application
- Application modifications and enhancements

Enhancing the Application

- Well-designed Java software minimizes the time required for:
  - Maintenance
  - Enhancements
  - Upgrades
- For Duke's Choice, it should be easy to:
  - Add new items to sell (business logic)
  - Develop new clients (presentation)
    - Take the application to a smartphone (for example)
  - Change the storage system (data)

Adding a New Item for Sale

It is possible to add a new item for sale by:

- Extending the Clothing or Camping class, or even creating a new category (for example, Books)
- Adding any new unique features for the item
- Adding some of the new items to the data store

Adding a New Item for Sale

Returnable is an interface and must be implemented.

```
8
9   duke.item.Suit is not abstract and does not override
10  abstract method doReturn() in duke.item.Returnable
11  --
    (Alt-Enter shows hints)
    public class Suit extends Clothing implements Returnable {
13
14  }
```

Suit is a type of Clothing.

Returns are permitted.

Implement Returnable

```
public class Suit extends Clothing implements Returnable {
    public String doReturn() {
        // In the current implementation Returnable provides
        // a marker that the item can be returned and also returns
        // a String with conditions for returning the item
        return "Suit returns must be within 3 days";
    }
}
```

Implement Constructor

```
public class Suit extends Clothing implements Returnable {

  ...< code omitted > ...

  // Types are D = Double-breasted, S = Single-breasted, U=Unset
  private char suitType = 'U'; //

  // Constructor
  public Suit(int itemID, String description, char colorCode,
              double price,  char type, int quantityInStock) {
    super( itemID,  description,  colorCode, price, quantityInStock);
    setSuitType(type);
    setSku(getSku() + type); // To create a unique SKU
  }
}
```

Suit Class: Overriding `getDisplay()`

```
  public String getDisplay(String separator) {

  String displayString = "SKU: " + getSku() + separator +
  "Item: " + getDescription() + separator +
  "Color: " + getColorCode() + separator +
  "Type: " + getSuitType() + separator +
  "Price: " + getPrice() + separator +

  "Available: " + getQuantityInStock();
  return displayString;
}
```

Implement Getters and Setters

```
public class Suit extends Clothing implements Returnable {

 ...< code omitted > ...

 public char getSuitType() {
   return suitType;
 }

 public void setSuitType(char suitType) {
   if (suitType!='D' && suitType!='B') {
      throw new IllegalArgumentException("The suit type must be"
                                  + " either D = Double-breasted "
                                  + "or S = Single-breasted");        }
        this.suitType = suitType;
   }
}
```

Updating the Applications with the Suit Class

For the command-line application:

- Create a new `DukesChoice.jar` file.
- (Optional) Copy it to a new location on the file system or to another machine.

For the web application:

- Create a new `DukesChoice.jar` file.
- Copy it to the directory that is used by the application server for library files.

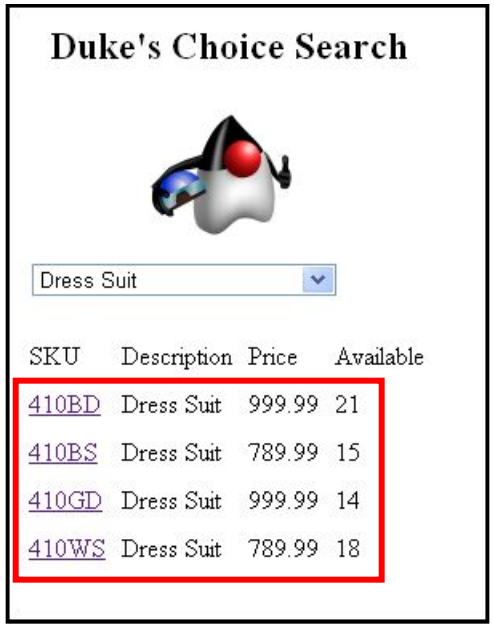
Testing the Suit Class: Command Line

```
C:\>java -jar
    "C:\work\Java_fundamentals\DukesChoice\dist\DukesChoice.jar"
    find 410
-----------------------------------------------------------------
SKU: 410BD | Item: Suit | Price: 999.99 | Color: B | Available: 21
-----------------------------------------------------------------
SKU: 410BS | Item: Suit | Price: 789.99 | Color: B | Available: 15
-----------------------------------------------------------------
SKU: 410gD | Item: Suit | Price: 999.99 | Color: G | Available: 14
-----------------------------------------------------------------
SKU: 410WS | Item: Suit | Price: 789.99 | Color: W | Available: 18
-----------------------------------------------------------------
```

Testing the Suit Class: Web Application

A new item appears in the drop-down menu.

The different kinds of suits added to the data store are listed.



Adding the Suit Class to the Web Application

The overridden `getDisplay()` method ensures that the suit type is displayed.

Summary

In this lesson, you should have learned how to:

- Deploy a simple application as a JAR file
- Describe the parts of a Java application, including the user interface and the back end
- Describe how classes can be extended to implement new capabilities in the application

No Practice for This Lesson

This lesson has no practices.

Course Summary

In this course, you should have learned how to:

- List and describe several key features of the Java technology, such as that it is object-oriented, multi-threaded, distributed, simple, and secure
- Identify different Java technology groups
- Describe examples of how Java is used in applications, as well as consumer products
- Describe the benefits of using an integrated development environment (IDE)
- Develop classes and describe how to declare a class
- Analyze a business problem to recognize objects and operations that form the building blocks of the Java program design

Course Summary

- Define the term *object* and its relationship to a class
- Demonstrate Java programming syntax
- Write a simple Java program that compiles and runs successfully
- Declare and initialize variables
- List several primitive data types
- Instantiate an object and effectively use object reference variables
- Use operators, loops, and decision constructs
- Declare and instantiate arrays and ArrayLists and be able to iterate through them

Course Summary

- Use Javadocs to look up Java foundation classes
- Declare a method with arguments and return values
- Use inheritance to declare and define a subclass of an existing superclass
- Describe how errors are handled in a Java program
- Describe how to deploy a simple Java application by using the NetBeans IDE