
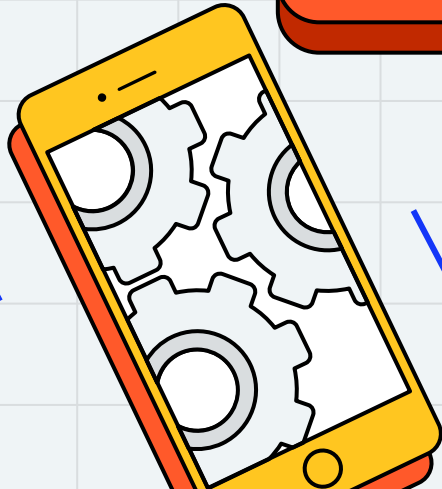


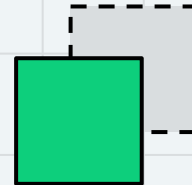


Application

Programming



Hend Alkittawi





More on Java Classes

Best practices for Creating Java
Classes

BEST PRACTICES FOR CREATING JAVA CLASSES

STYLE



- identifier naming
- indentation

METHODS



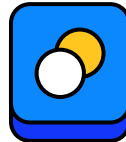
- getters and setters
- equals()
- toString()

CONSTRUCTORS



- Ways to instantiate objects

JAVADOC



- "special" Java documentation

STYLE

```
visibility  type  class name
↓          ↓          ↓
public class MyClass {
    // attributes
    private int intVar;
    ↑      ↑      ↑
visibility type variable name
// methods
public void myMethod(String myString){
↑      ↑      ↑      ↑      ↑
visibility return method parameter parameter
}      type  name  type  name
}
```

METHODS

- The syntax (rules) for declaring a **method**

visibility **returnType** **methodName**(**parameterList**)

- **visibility** determines access
 - public (all can access) or private (just this class can access)
- **returnType** is the type of data this method returns
 - if nothing is returned, use the keyword void
- **methodName** starts with a lowercase word
 - it uses uppercase for the first letter of each additional word (this is called "camel case")
- **parameterList** is any data we need to pass to the method
 - The ordering must be followed exactly

METHODS

- Getter and setter methods are used to get and set the **value of the attributes**.
 - Getters retrieve the value only
 - Setters update the value only

```
public class Account {  
  
    private double balance;  
    private String name;  
  
    public void setBalance( double balance ){  
        this.balance = balance;  
    }  
    public double getBalance(){  
        return this.balance;  
    }  
    // add a getter and a setter for name  
}
```

- to use the class and its methods

```
public static void main(String[] args){  
    Account myAccount = new Account();  
    myAccount.setBalance(1000.0);  
    double balance = myAccount.getBalance();  
}
```

METHODS

- `toString()` method returns a String representation of objects.

An example of a `toString()` method for the Account class:

```
public class Account {
    private double balance;
    private String name;
    public void setBalance( double balance ){
        this.balance = balance;
    }
    public double getBalance() {
        return this.balance;
    }
    // ... getter and a setter for name
    public String toString() {
        return "Account info: name: " + name
            + " with balance: " + balance;
    }
}
```

- to use the class and its methods

```
public static void main(String[] args){
    Account myAccount = new Account();
    myAccount.setName("Sam Adams");
    myAccount.setBalance(1000.0);
    System.out.println( myAccount );
}
```

METHODS

- `equals()` method provides a way for users to compare instances of your object to other instances. This also gives you control over what is relevant to differentiate your objects.

```
public class Account {  
  
    private double balance;  
    private String name;  
  
    public void setBalance( double balance ) {  
        this.balance = balance; }  
  
    public double getBalance() {  
        return this.balance; }  
  
    public String toString() {  
        return "Account info: name: " + name  
            + " with balance: " + balance; }  
  
    public boolean equals( Account account2 ) {  
        return this.getName().equals( account2.getName()); }  
}
```

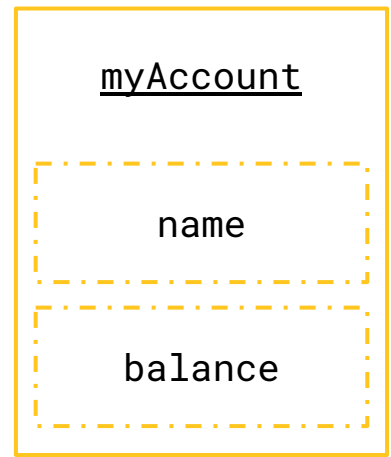
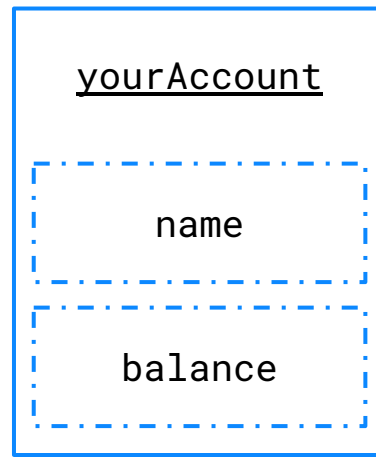
- to use the class and its methods

```
public static void main(String[] args){  
    Account myAccount = new Account();  
    Account yourAccount = new Account();  
    myAccount.setName("Mia");  
    myAccount.setBalance(10);  
    yourAccount.setName("Ken");  
    yourAccount.setBalance(100);  
    boolean check = myAccount.equals(yourAccount);  
}
```


IMPORTANT

- `==` is used for primitive types only
 - `2 == 5`
- Objects define an object-method called `equals()`
 - `objA.equals(objB);`
- [Core: Drawing Memory Models with Objects - Memory Models, Scope, and Starting the Project | Coursera](#)

```
public class Account {  
  
    private double balance;  
    private String name;  
  
    public void setBalance( double balance ) {  
        this.balance = balance; }  
  
    public void setName( String name ) {  
        this.name = name; }  
  
    public double getBalance() {  
        return this.balance; }  
  
    public String getName() {  
        return this.name; }  
  
    public boolean equals( Account account2 ) {  
        return this.getName().equals( account2.getName());  
    }  
  
    public String toString() {  
        return "Account info: name: " + name  
            + " with balance: " + balance; }  
}
```



```
public static void main(String[] args){  
    Account myAccount = new Account();  
    Account yourAccount = new Account();  
    myAccount.setName("Mia");  
    myAccount.setBalance(10);  
    yourAccount.setName("Ken");  
    yourAccount.setBalance(100);  
    boolean check = myAccount.equals(yourAccount);  
}
```

CONSTRUCTORS

- Java requires a constructor call for every object that is created.
- The keyword `new` creates a new object of a specified class by calling a constructor.
- A constructor is similar to a method but is called implicitly by the `new` operator to initialize an object's instance variables when the object is created.
- In any class that does not explicitly declare a constructor, the compiler provides a `default constructor` (which always has no parameters).
- When a class has only the default constructor, the class's instance variables are initialized to their default value.

METHODS

- You can provide your own constructor to specify custom initialization for objects of your class.
- A constructor must have the same name as the class.
- If you declare a constructor for a class, the compiler will not create a default constructor that class.

```
public class Account {  
  
    private double balance;  
    private String name;  
  
    public Account( String name, double balance ){  
        this.name = name;  
        this.balance = balance;  
    }  
    /* ... rest of class ... */  
}
```

- to use the class and its methods

```
public static void main(String[] args){  
    Account myAccount = new Account("Sam", 1000);  
    Account yourAccount = new Account("Jane", 2000);  
}
```

JAVA GARBAGE COLLECTION

- More than one variable may refer to the same data.
- Java will clear out old data that no variables are referencing
 - This is known as garbage collection
- **Garbage collection** is the process through which Java will eventually clear out old data that no variables are referencing.

JAVADOC

- Javadoc comments are delimited between `/**` and `*/`.
- Javadoc comments enable you to **embed program documentation** directly in your programs.
- Javadoc utility program reads Javadoc comments and uses them to prepare program documentation in HTML web-page format.

```
/**
```

```
 * This is a Javadoc comment!
```

```
*/
```

JAVADOC

- Javadoc comments annotations:
 - `@author`: designates the author of the code, belongs in the class comment
 - `@param`: designates the parameter to a method, belongs in all method comments which require parameters
 - `@return`: designates the returned value of a method, belongs in method comments which return values

JAVADOC

Javadoc

```
/**  
 * The Account class represents a bank account.  
 * @author CS3443  
 */
```

```
public class Account {
```

```
    /* attributes here .. a multi-line comment starts with /* .. be careful! */
```

```
    /**  
     * sets the account balance  
     * @param balance, the account balance (double)  
     */
```

```
    public void setBalance(double balance){  
        this.balance = balance;  
    }
```

```
    /**  
     * returns the balance for the account  
     * @return double, the account balance  
     */
```

```
    public double getBalance(){  
        return balance;  
    }
```

```
}
```

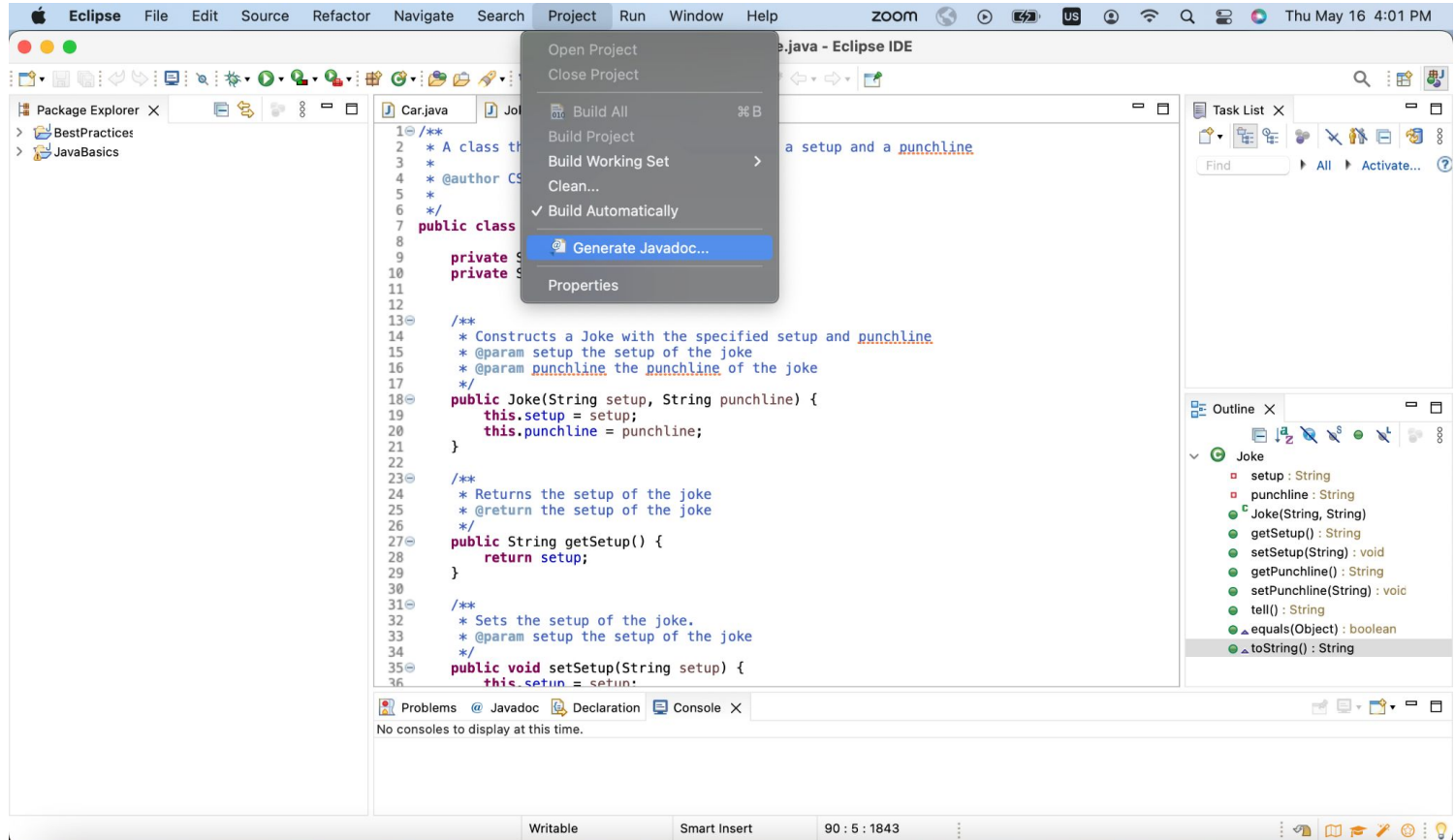
Multiline
Comment

Javadoc

JAVADOC

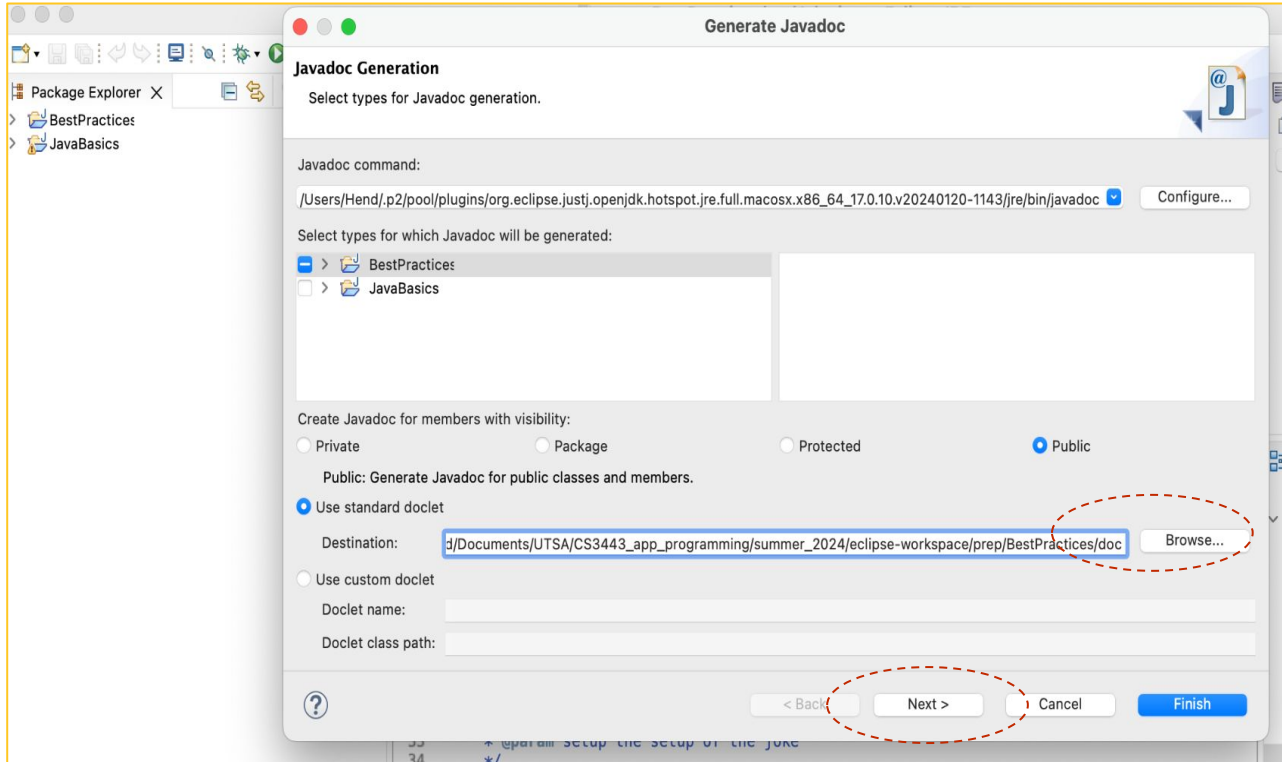
- To generate Javadoc in Eclipse
 1. Project > Generate Javadoc
 2. Destination: workspace/your_project/doc
 3. Next
 4. Select all "referenced archives and projects"
 5. Finish > Yes To All
 6. Open index.html

Project > Generate Javadoc



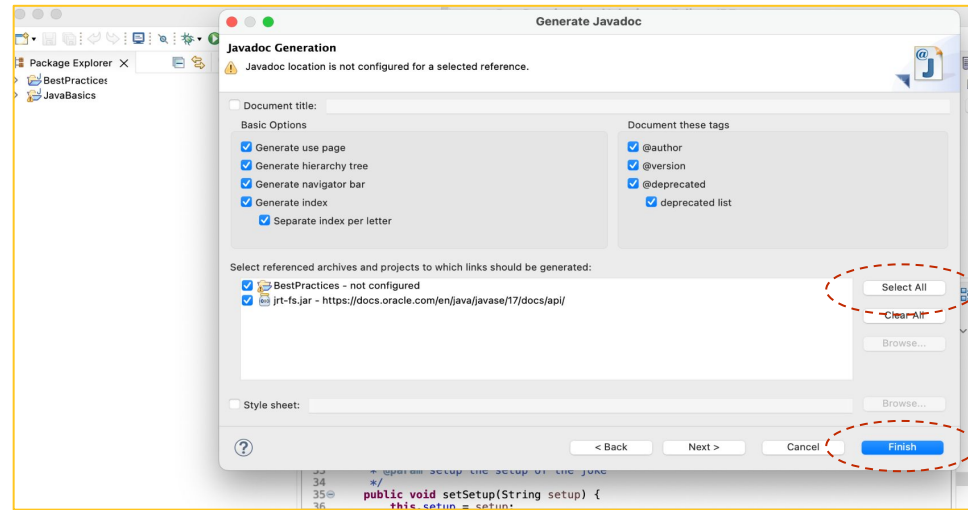
Destination:workspace/your_project/doc

> Next

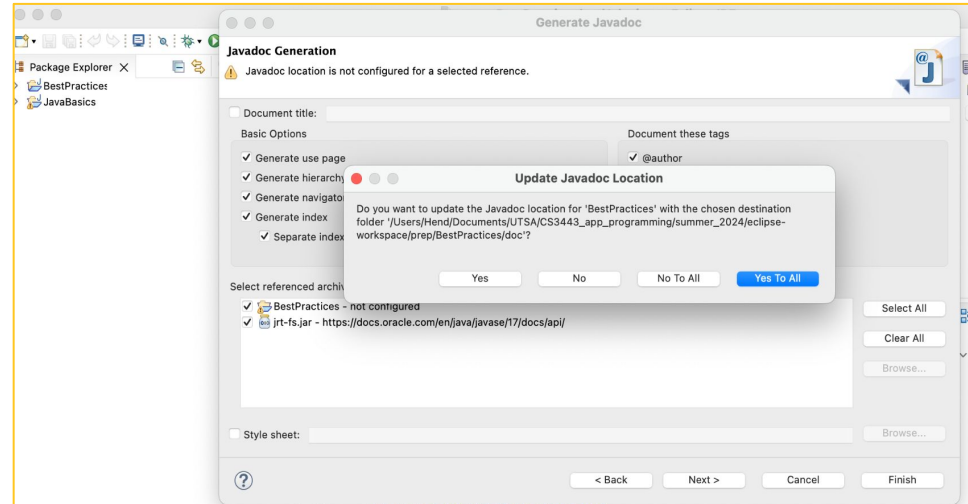


Select all

> Finish



Yes To All



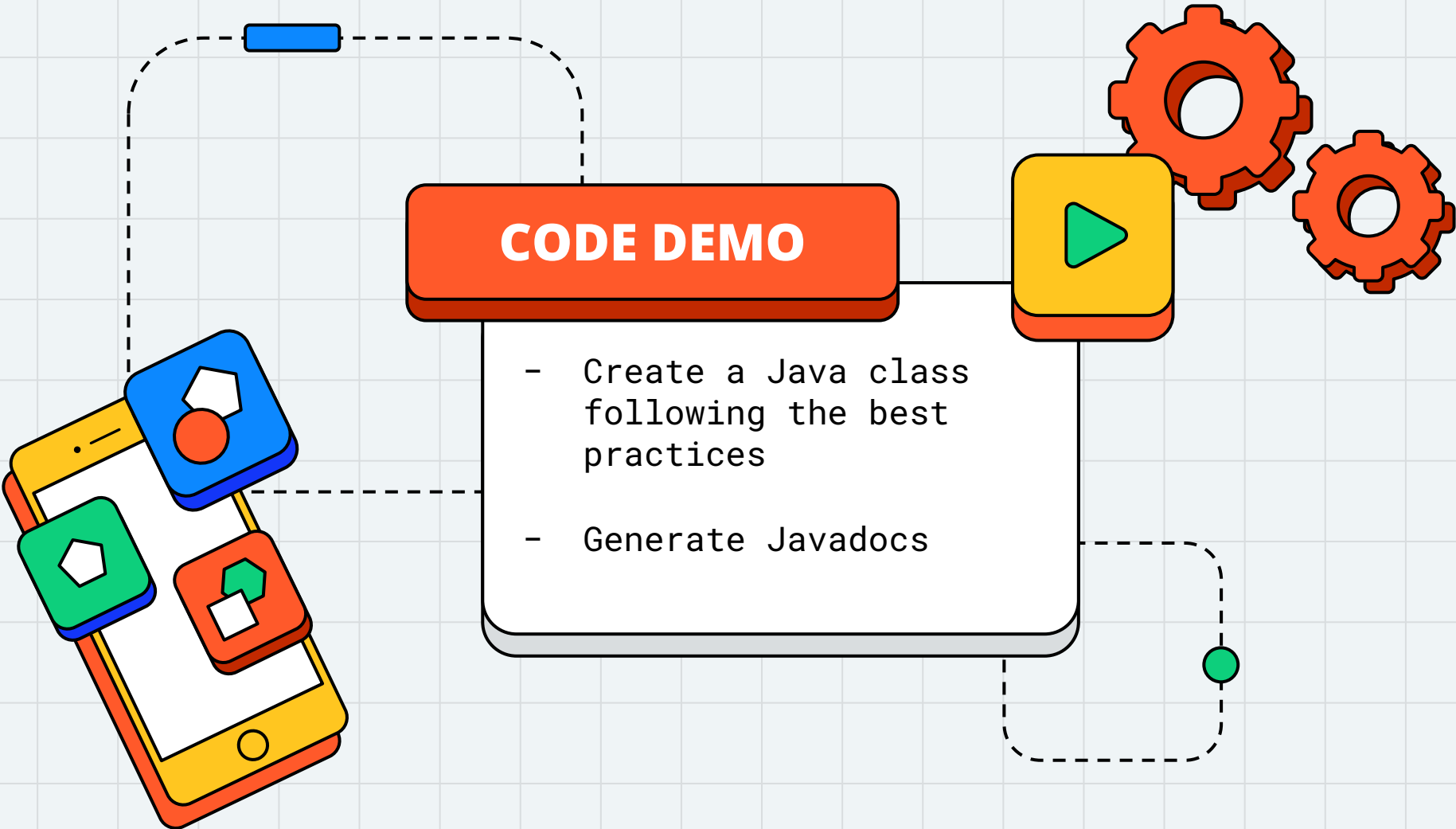
Open index.html

The screenshot shows the Eclipse IDE interface with the following components:

- Package Explorer:** Located on the left, it shows a project named "BestPractices" with a sub-package "src". Under "src", there is a folder "doc" containing several HTML files. The file "index.html" is highlighted with a red dashed circle.
- Editor:** The main window displays the source code for "Joke.java". The code is as follows:

```
1 1 /*  
2 2  * A class that represents a story joke with a setup and a punchline  
3 3  *  
4 4  * @author CS3443  
5 5  *  
6 6  */  
7 7  public class Joke {  
8 8  
9 9      private String setup;  
10 10     private String punchline;  
11 11  
12 12  
13 13     /**  
14 14     * Constructs a Joke with the specified setup and punchline  
15 15     * @param setup the setup of the joke  
16 16     * @param punchline the punchline of the joke  
17 17     */  
18 18     public Joke(String setup, String punchline) {  
19 19         this.setup = setup;  
20 20         this.punchline = punchline;  
21 21     }  
22 22  
23 23     /**  
24 24     * Returns the setup of the joke  
25 25     * @return the setup of the joke  
26 26     */  
27 27     public String getSetup() {  
28 28         return setup;  
29 29     }  
30 30  
31 31     /**  
32 32     * Sets the setup of the joke.  
33 33     * @param setup the setup of the joke  
34 34     */  
35 35     public void setSetup(String setup) {  
36 36         this.setup = setup;  
37 37     }  
38 38 }
```
- Task List:** Located on the right, it is currently empty.
- Outline:** Located on the right, it shows the class structure for "Joke":
 - setup : String
 - punchline : String
 - Joke(String, String)
 - getSetup() : String
 - setSetup(String) : void
 - getPunchline() : String
 - setPunchline(String) : void
 - tell() : String
 - equals(Object) : boolean
 - toString() : String
- Problems Console:** Located at the bottom, it shows the output of the Javadoc generation process:

```
<terminated> Javadoc Generation  
Loading source file /Users/Hend/Documents/UTSA/CS3443_app_programming/summer_2024/eclipse-workspace/prep/BestPractices/src/Joke.jav  
Constructing Javadoc information..  
Building index for all the packages and classes...  
Standard Doclet version 17.0.10+7  
Building tree for all the packages and classes...
```





THANK

YOU!

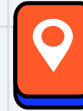
DO YOU HAVE ANY QUESTIONS?



hend.alkittawi@utsa.edu



By Appointment



Online