**OBJECT-ORIENTED PROGRAMMING (OOP)**

* Object Oriented is a programming **paradigm**
* Paradigm is **style of programming**



**Benefits of Object-Oriented Programming**

* Object Oriented programming is about objects
* These objects interact with one another to perform various tasks
* As an analogy let’s think of a car
* Cars consist of many collaborating objects
* Almost all of the objects are re-usable or replaceable 

**Course structure:**



**Classes and Objects**

* Class = a blueprint/template for creating objects 
* Object = an instance of class
* From the above example CAR is a class and car1,car2 and car3 are objects

**Creating Classes**

* As before we create a brand new project like HelloWorld
* From here we go to the SRC folder and right click on this to create a separate class



**Remember that naming of a class we use Pascal naming convention with each word starting with a capital**

* We add 3 members within the curly braces namely
  + Field = Public access modifier, then specify the type of this field, and then give it a name (using camelCasing notation) NOTE: As we become more experienced we will know that we do not define fields as public as is done below
  + Let’s declare a couple of methods  
    NOTE: While declaring the method we note that the name of the parameter is exactly the same as the name of the field So we use the this keyword to point to the field  
      
      
      
    

**Creating Objects**

* We will declare this in our main method
* We declare a TextBox variable so call it textBox1 and initialise it as follows 
* We use the new operator to create a new instance of the TextBox class
* NOTE: We are instantiating the TextBox class which means we are creating a new instance of this class
* NOTE: We can make our code even cleaner by not repeating TextBox. We can simplify our code by using the var keyword   
    
    
  NOTE: The Java compiler will detect the type of this variable based on what we have on the right side of the assignment operator(=)
* Next we use the dot operator to check the members of this class   
    
  

**Encapsulation**

* This is the first principal of OOP.
* **Encapsulation** = Bundle the data and methods that operate on the data in a single unit/object
* We can use the encapsulation principal by creating a class  

**Getters and Setters (REVISIT).**

* In Java we use setters to set the value of a field and getter to get the value of a field
* We need to create a method whereby we do some data validation
* Here is an example of the method for a setter and note that the access modifier once we do this will be private in our new Employee class  
    
  
* Here is an example of a setter  
* If we wanted to get the value of the baseSalary field we can use a getter. See below  
   
* NOTE: Very important trick is hover on a field that you want to create a getter/setter and press [alt] + [ENTER] and this will appear. This short cut will have Intellij suggesting to automatically create a getter and setter as we have done before.

**Abstraction**

* This is the second principle of object-oriented programming
* Abstraction = We should reduce complexity by hiding unnecessary details in our classes
* With abstraction we want to hide the implementation detail of a class and treat it like a black box .i.e buttons of a remote control
* Because the fields below are like the transistors in a remote control we made them private. We do not want to work to work directly with these fields.

**Coupling**

* Coupling = The level of dependency between classes
* We need to try to reduce coupling between classes
* The benefit of this is so that once a class is changed then the dependant class need to be modified. The more classes there are dependent then we will have to modify more.
* Buttons on a remote are like different methods

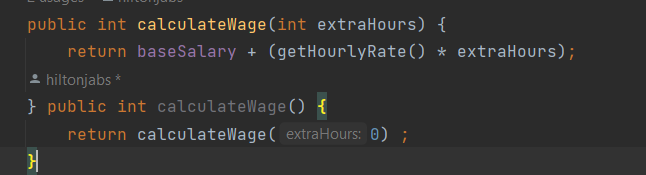
**Reducing Coupling**

* Refer to the Browser class example

**Constructors**

* We can use default constructor to initialise our fields to default values i.e numbers will be zero, Booleans will be false and reference types are going to be null
* We constructors to initialise our objects

**Method Overloading**

* Creating different implementations of a method but with different parameters
* Shortcut CTRL + D will repeat the highlighted code
* Below example is of overloading the calculateWage method twice, in one implementation we get the extra hours and in the other implementation we do not receive values
* Not recommended : be aware of method overloading but don’t have to use it as there are minimal advantages

**Constructor Overloading**

* ShortCut: Jumping from one method in the main to the class use CTRL + B (Declaration) this can also be found in the navigate menu
* Another shortcut is under VIEW, PARAMETER info you can view where the parameter has been used.

**Static Members**

* Static Members = Static Fields and Static methods
* In object-oriented programming a class can have 2 types of members
  + Instance members
  + Static members

**Revision Quiz**

**Questions**

1. **What is the difference between a class and an object?**   
     
   A class is a blueprint or template for creating objects. An object is an instance of a class.

2- **What does instantiating mean?**   
  
Instantiating means creating an instance of a class: new Customer()  
  
  
3- **What is the difference between stack and heap memory? How are they managed?**   
Stack is used for storing primitive types (numbers, boolean and character) and variables that store references to objects in the heap. Variables stored in the stack are immediately cleared when they go out of scope (eg when a method finishes execution). Objects stored in the heap get removed later on when they’re no longer references. This is done by Java’s garbage collector.   
  
4- **What are the problems of procedural code? How does object-oriented programming help solve these problems?**   
  
Big classes with several unrelated methods focusing on different concerns and responsibilities. These methods often have several parameters. You often see the same group of parameters repeated across these methods. All you see is procedures calling each other passing arguments around. By applying object-oriented programming techniques, we extract these repetitive parameters and declare them as fields in our classes. Our classes will then encapsulate both the data and the operations on the data (methods). As a result, our methods will have fewer parameters and our code will be cleaner and more reusable.   
5- **What is encapsulation?**   
  
Encapsulation is the first principle of object-oriented programming. It suggests that we should bundle the data and operations on the data inside a single unit (class).  
  
6- **Why should we declare fields as private?**   
  
- How we store data in an object is considered an implementation detail. We may change how we store the data internally. Plus, we don’t want our objects to go into a bad state (hold bad data). That’s why we should declare fields as private and provide getters and or setters only if required. These setters can ensure our objects don’t go into a bad state by validating the values that are passed to them.   
  
7- **What is abstraction?**

Abstraction is the second principle of object-oriented programming. It suggests that we should reduce complexity by hiding the unnecessary implementation details. As a metaphor, think of the remote control of your TV. All the complexity inside the remote control is hidden from you. It’s abstracted away. You just work with a simple interface to control your TV. We want our objects to be like our remote controls.

8- **What is coupling?**

Coupling represents the level of dependency between software entities (eg classes). The more our classes are dependent on each other, the harder it is to change them. Changing one class may result in several cascading and breaking changes.  
  
9- **How does the abstraction principle help reduce coupling?**   
  
By hiding the implementation details, we prevent other classes from getting affected when we change these details. For example, if the logic board and transistors inside a remote control change from one model to another, we’re not affected. We still use the same interface to work with our TV. Also, reducing these details and exposing fewer methods makes

our classes easier to use. For example, remote controls with fewer buttons are easier to use.

10- **What are constructors?**By hiding the implementation details, we prevent other classes from getting affected when we change these details. For example, if the logic board and transistors inside a remote control change from one model to another, we’re not affected. We still use the same interface to work with our TV. Also, reducing these details and exposing fewer methods makes

our classes easier to use. For example, remote controls with fewer buttons are easier to use.

11- **What is method overloading?**   
  
Method overloading means declaring a method with the same name but with different signatures. The number, type and order of its parameters will be different..  
  
  
12- **What are static methods?**

Static methods are accessible via classes, not objects.