**Section 6\_OOP Part1 Classes, Constructors**

Topics Covered:

Classes

Constructors

IntelliJ short cut

Inheritance

**Classes**

Same as C++

How to make classes in IntelliJ

* Make a project as normal
* Drop down menu classes -> src -> right click on package name or on src -> new -> java class

Fields members of a class

Example

public class Car  
{  
 // the car class also inherits some function by default from the base java class  
 private int doors;  
 private int wheels;  
 private String model;  
 private String engine;  
 private String colour;  
  
 public void setModel(String model){  
 String validModel = model.toLowerCase();  
  
 if(validModel.equals("carrera") || validModel.equals("commodore")){  
 this.model = model;  
 }  
 else {  
 this.model = "unknown";  
 }  
 }  
  
 public String getModel(){  
 return model;  
 }  
}

public class Main {  
 public static void main(String[] args) {  
 Car porsche = new Car();  
 Car holden = new Car();  
 porsche.setModel("C");  
 System.*out*.println("The car model is " + porsche.getModel());  
 }  
}

**Constructors**

Same as C++

**NB:** some suggest nit calling setters in constructors as it sometimes causes problem (it is a conflicting opinion)

public class Car  
{  
 // the car class also inherits some function by default from the base java class  
 private int doors;  
 private int wheels;  
 private String model;  
 private String engine;  
 private String colour;  
  
 public Car(){  
 //calling one constructor from another constructor  
 this("porse", "pink");  
 System.*out*.println("Empty Constructor called");  
 }  
  
 public Car(String model, String colour){  
 this.model = model;  
 this.colour = colour;  
 }  
  
 public Car(int doors, int wheels, String model, String engine, String colour) {  
 this.doors = doors;  
 this.wheels = wheels;  
 this.model = model;  
 this.engine = engine;  
 this.colour = colour;  
 }  
  
 public void setModel(String model){  
 String validModel = model.toLowerCase();  
  
 if(validModel.equals("carrera") || validModel.equals("commodore")){  
 this.model = model;  
 }  
 else {  
 this.model = "unknown";  
 }  
 }  
  
 public String getModel(){  
 return model;  
 }  
  
 public String getColour() {  
 return colour;  
 }  
  
 public void setColour(String colour){  
 this.colour = colour;  
 }  
  
}

**IntelliJ short cut**

Code -> generate -> constructor -> (shift and hover mouse to select) -> okay

**Inheritance**

public class Dog extends Animal{

The rest is the same as C++

Super calls the constructor that is for the class that we are extending from

Always use super when calling parent methods from child class

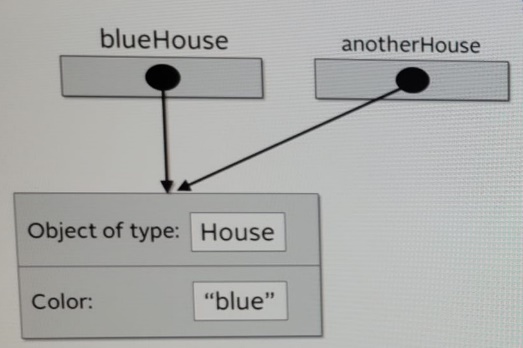
public void walk(){  
 System.*out*.println("Dog is walking");  
 super.eat();  
}

**Reference**

Address of an object in memory

House blueHouse = new House(“blue”);

House anotherHouse = blueHouse;



House greenHouse = new House(“green”);

anotherHouse = greenHouse;

Diagram

Description automatically generated

**NB:** In Java you always have reference to an object in memory, there is no way to access an object directly everything is done using reference.

**This vs super**

Super it is used to access or call parent class members (variables and methods)

This it is used to call the current class members (variables and methods).

This is required when we have a parameter with the same name as an instance variable (field)

public Car(){  
 //calling one constructor from another constructor  
 this("porse", "pink");  
 System.*out*.println("Empty Constructor called");  
}  
  
public Car(String model, String colour){  
 this.model = model;  
 this.colour = colour;  
}

**Nb:** we can use both anywhere in a class except static areas (the static block or static method). Any attempt to do so will lead to compile-time errors (more on static later in the course)