Objects and Classes

* **OOP :** Object Oriented Programming involves programming using objects. For example, a student, a desk, a circle, a button and even a Lion can all be viewed as objects.

* An object has a unique identity, state, and behaviors.
state ↓ properities — behaviors ↓ methods

* Constructors

→ A special kind of methods that are invoked to construct objects

* defult or no argument constructor. →

* Con. must have the same name as the class itself.

* Con. do not have a return type — not even void.

```
public Circle() {
}

public Circle(double newRadius) {
    radius = newRadius;
}
```

→ Con. are invoked using the new operator when an object created. Examples :

new Circle(); defult من النوع الأول ال
new Circle(5.0); من النوع الثاني
التي تستقبل double .

note : إذا ما عملنا ال Constructor ال IDE بعمل default Con. تلقائي .

* Example of declaring objects :

```
Circle mycircle = new Circle();
```

- `mycircle =` → Assign object reference.
- `new Circle()` → Creating an object

* Accessing Object's Members

→ objectReferenceVariable . data → عن طريق النقطة

e.g : myCircle .radius ;

objectRevVar . methodName(arguments)

eg: myCircle. getArea() ;

* Default value for a Data Field :

null → strings

0 → numbers

false → boolean

'\u0000' → char

However, Java assigns no defult value to a local variables inside a method.

Example : inside main method ⇒ int x ;

String Y ;

System.out.print ("x is" + x);

" " " ("Y is" + Y);

Compilation error

* بس لو X , Y من ثاني Class

حتى لو انا مش معطيهم قيم يطبع 0 , null

* Date class :

```
java.util.Date date = new java.util.Date();
System.out.print ( date. toString () );
```

→ Mon Mar 20 15:42:52 IST 2023

* Random Class

→ Math.random() ;

→ java.util.Random class

(variable and Method)

* Instance ↑ belongs to [redacted] object.

* Static (variable, costant and Method) belongs to Class.
  → Variable : initialized only one befor any Instance Variable
  Single Copy
  accessed directly by the class name
  (class_name). (static variable name)
  → Method : Can access only static data
  Can call only other static method
  accessed directly by the class name.
  (class_name). (static_method_name)
  note: main method is static.
  Can not use "this" or "super". (we use name_class)

* Visibility Modifiers :
  Private modifier restricts access to within a class.
  Default " " " " " a package.
  Public modifier enables unrestricted access.

* Why data Fields should be private?
  To protect data.
  To make code easy to maintain.
  * Encapsulation proccess *

* Overloading
  several methods with the same name. However, they must have different signature (parameter types and the order of parameter).
  If we have two methods have different return type (int, double) or visibility (public, private) or throwing exceptions, they don't overloading.

* Passing Objects to Methods: ← بعطي الجنور نسخة وبتفضل قيمته في الميد زي ماهي
primitive type → Pass by value.
reference (object) type → Pass by reference. بعطي الجنور القيمة الحقيقية وبتتغير في ال main كمان.

* Array of objects :
→ In Object
Circle myCircle = new Circle(); Creating object
→ In Array of Objects
Circle[] arrayCircle = new Circle[10]; Creating ~~refer~~ array of references.
arrayCircle[0] = new Circle(); creating object

* Immutable Class :
① All data fields private.
② No setters methods.
③ No getters methods that return a reference.

* Scope of Variables :
→ In Instance (object) and static variables can be declared anywhere inside a class
→ local variables starts from its decleration to the end of the block. And, ~~must~~ it must be initialized.

* this Keyword
It's a reference that refers to an object itself.
we use "this" → to reference a class's hidden data field.
↳ to invoke another constructor of the same class.
→ calling overloded constructor :

```
public class Circle {
  private double radius;
  public Circle (double radius) {
  this.radius = radius;
  }
  public Circle() {
  this (1.0);
  }
  public double getArea() {
    return this.radius * this.radius * Math.PI;
  }
}
```

وجودهم وعدمهم واحد

الاستخدام الأول

الاستخدام الثاني

Inheritance and Polymorphism

* The Subclass Inherited properites and methods from the Superclass and invoked the constructor (not Inherited):
  - → Explicitly with using "Super" Keyword.
  - → Implicitly without using "Super" Keyword, as an overloaded constructor or a super class's constructor.

  note: If none of them invoked explicitly, the Compiler puts "super()" as the first statement in the constructor.

* Super uses:
  To call a superclass constructors.
  To call a superclass methods.

* Caution:
  we use "Super" instead of superclass constructor name.
  we must put the "Super" in the first statement of constructor.

* A Subclass inherits from Superclass and you can:
  - → Add new properties
  - → Add new methods
  - → Override the methods of the superclass.

* Overriding methods (يُغيّر)
  The subclass modify the implementation of a method defined in the superclass.
  - → we can overriding method only if it is accessible.

  So, Private cannot be overriden.

وكان في زيها في Superclass

→ If we defined a private method in Subclass, the two methods are unrelated.

→ Static method can be inherited, but cannot be overriden. Also, static method redefined in subclass and is hidden.

*

في الجافا كلاس اسمه (Object) أبو الجميع

إذا عملت منه (extends) أو لا تغير النتيجة

← في هذا الكلاس توجد ميثود toString()

إذا بدنا نستخدمها مع أي متغير راح تطبع Classname @ 12345e

ولكن هذا استخدام بلا معنى.

* نستفيد منها عن طريق @ override ← بتطبع اسم الكلاس وكل المتغيرات والميثودز.

ولما يورثها subclass منتقدم Suber. toString() ومنضيف كمان متغيرات لـ subclass.

ويرضى بعمل override من الـ Superclass اللي أريده عمل override من الـ Object.

* Polymorphism تعدد الأشكال

An object of a subtype can be used wherever its Supertype value is required.

```
public class Demo {
    public static void main(String[] a) {
        m(new Object());
        m(new Person());
        m(new Student());
        m(new GradnateStuden());
    }
    public static void m(Object X) {
        sys.out.print(X.toString());
    }
}
```

في كل مرة بعوض object مختلف راح يرجع معاه toString الخاصة فيه. ← هذا يسمى Dynamic Binding

↳ يعني toString منو معروفة شو هي الا وقت runtime

(Poly.) بمعلنا نستخدم الميثودز بشكل واسع في الـ arguments

هذا يسمى generic programming

Casting Objects

used to convert an object of one class type to another within Inheritance.

→ In previous Example :

m(new Student()); ≡ [Object o = new Student() ; (implicit casting)
m(o) ;]

Known as implicit casting, because an istance of Student is automatically an instance of Object

⇒ Student b = o ; // Compile error

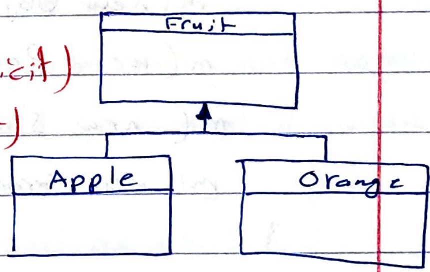
* why does Object o = new Student() work and Student b = o doesn't ?
Because Student is always instance of Object, but an Object is not necessarily (ليس بالضرورة) an istance of Student.

⇒ To fix the error : Student b = (Student) o ; // Explicit casting

→ Explicit casting must used when casting from Superclass to subclass. And, may not always succeed.

مثال شامل →

```
Fruit fruit = new Apple(); (implicit)
Apple a = (Apple) fruit ; (explicit)
Orang o = (Orange) fruit; ✗ error
```



* عشان ما نوقع بمشاكل ال Casting في عنا شغلتين بريحونا :

① instanceof opertor ② equals method (Object موجودة في)

مثال بشرحهم → For example, the equals method is overriden in the Circle class:

```
public boolean equals (Object o) {
    if (o instanceof Circle)
        return radius == ((Circle) o . radius) ;
    else
        return false ;
}
```

→ بقدر أقارن بأكثر ثقة
مش شرط radius.

note:

"==" → to compare two primitive data type or same reference.

"equals" → to test two objects have the same value (contents).

* Protected Modifier:

→ Protected modifier can be applied on data and methods.

→ Protected data/method in public class can be accessed by any class in the same package or its subclasses, even if the subclasses are in a different package.

note: we cannot use protected with class

* Visibility increases →

private, none (defualt), protected, public

* Subclass cannot weaken the Accessibility

الـ Subclass ما بقدر يضعّف الوصول يا بخلّيه زي ما هو أو بزيده.

* final Modifier

final class cannot be extended آخر السلسلة ما بقدر حد يورث منّه

Ex: final class Math

final variable is a constant

Ex: final static double PI = 3.14159 ;

final method cannot be overriden by its subclasses

note: جميع الـ modifiers منستخدم في الـ class and class members

ما عدا الـ (final) منقدر نستخدمها كـ (local variable) داخل الميثود.

✳ The ArrayList Class

Java provides the ArrayList class that can be used to store an unlimited number of objects.

→ Creating ArrayList :

```
ArrayList<String> cities = new ArrayList<String>();
or ArrayList<String> cities = new ArrayList<>();
```

✳ Differences between Arrays and ArrayList :

operation	Array	ArrayList
Creating Array/AL	String[] a = new String[10]	نفس فوق
Accessing an element	a[index]	list.get(index);
updating an element	a[index] = "London";	list.set(index, "London");
Returing size	a.length	list.size();
Adding a new element	X	list.add("London");
Inserting a new element	X	list.add(1nder, "London");
Removing an element	X	list.remove(index);
Removing an element	X	list.remove(object);
Removing all element	X	list.clear();

✳ Creating ArrayList from Array

```
String[] array = {"red", "green", "blue"};
⇒ ArrayList<String> list = new ArrayList<>(Arrays.asList(array));
```

✳ Creating Array of objects from ArrayList

```
String[] array1 = new String[list.size()];
list.toArray(array1);
```