CH

Abstract classes

* يخدمني عندما توجد فراغات في الأب ، لا يمكنني الاستفادة منها داخل الابن
* توفر عناء الخوض في Down Casting و instanceof .

ex Suppose I have these classes:
Shape , Triangle , Rectangle, Driver

* Triangle & Rectangle have getArea method
لا يمكنني تنفيذ الحال وهو getArea داخل shape لانه لا يعرف كيف يحسبها

in main
```
Shape a= new Rectangle();
Shape b= new Triangle();
```

Same Area

in Driver Public static boolean (Shape a, Shape b){

احتاج ان اعرض احتمالات عِدّة لكل حالة باستخدام
DownCasting instance of

ما الحل ؟
نعرّف abstract لـ Shape

⇒ Public abstract class Shape{
String Color;

⇒ Public abstract getArea();
}
هكذا يفهم انه سيأخذ المساحة من الابن
الذي هو actual type :

So :
in Driver :
public static sameArea (Shape a, shape b){
~~if (a~~ return a.getArea == b.getArea(); }
لا احتاج DownCasting ← لانه a و b كل واحد ~~[illegible]~~
يحتوي على getArea معرفة في Shape
∴ no errors

لا احتاج ~~DownCasting~~
instance of ← لانه a,b يذهب لا يبحث actual type

عند المساحة الفرعية

Note if there is an abstract method
⇒ the class must be astract

if the class is abstract
⇒ the method may be abstract (or may not)

Note | I Can not create an instance (object) from abstract class

⇒ Shape s = new Shape(); ✗

لكن إلي فيه constructor لأنه بيستعمل الـ Initialization

Note | abstract class is allowed to be declared type but not actual type.

---

* Shape [] shapes = new Shape [3]
→ allowed → subtype داخلها كل واحد يكون من نوع

---

if subclasses of Shape didn't implement the getArea method ⇒ the sub. must be abstract

Rule ⇒ in General:
if I have an abstract class, contains some abstract methods.
the subclasses must implement All abstract methods in his super class or the subclass must also be abstract then.

abs. ابن الابن لازم يحتوي الـ abs. الي في الاب عالجد وكل مما فوقه
والا لازم يكون هو abs.

Notes // Concrete (not abstract) Super class may have abstract subclasses.

// abstract super class ~ ~ concrete ~

// ~ ~ ~ ~ ~ abstract ~

// Concrete ~ ~ ~ ~ Concrete ~

* Abstract in UML ⇒ مكتوبة بخط مائل

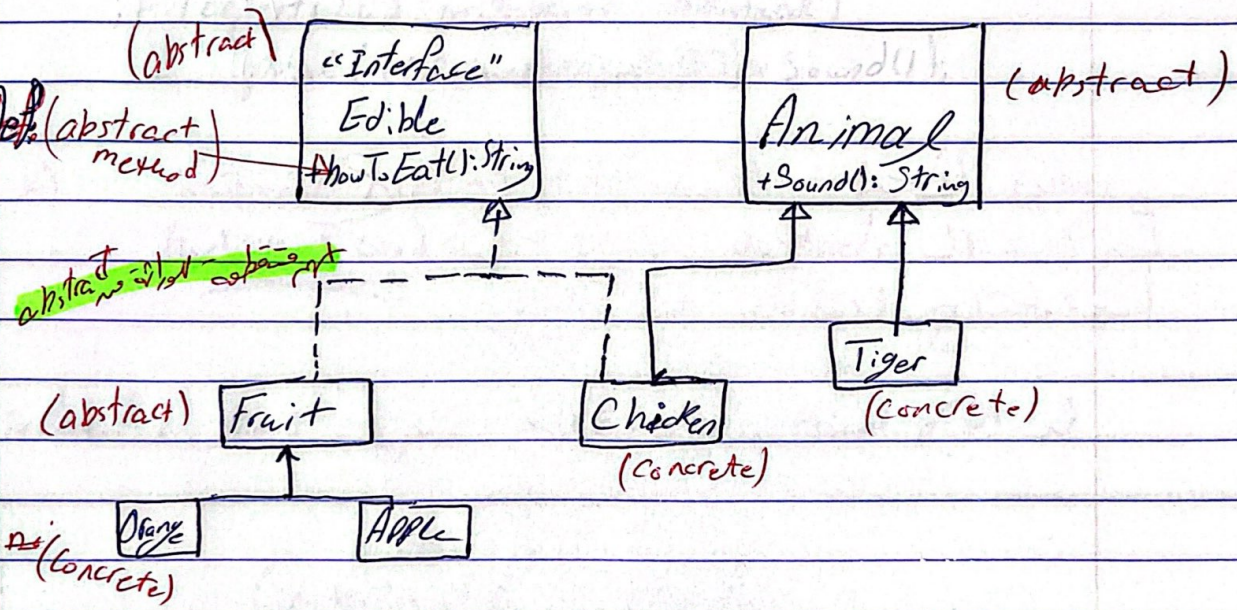
So.. abstract classes may have Variables, constans concrete methods, abstract methods

Interfaces

Interfaces have only constants and abstract methods

it is 'Weak is-a relationShip'.

تعبير عن تعريف معين



*) حتى لازم ابناء interface يكونوا abstract ، عكس
Constants + abs. methods لكن الواجب ان يكونوا

*) Fruit is abstract since it did not implement all the super class abstract methods

*) interface ← الدائرة من : UML

*) Use KeyWord 'implements' to extends interfaces

ex → Public class Chicken extends Animal implements Edible{

→ abstract class Fruit implements Edible{

*) لأنك ممكن تورث من أكثر من مكان ⟸ نستخدم interfaces

TestEdible:

```
Object [] objects={new Tiger(), new Chicken(), new Apple()};
for(int i=0; i< objects.length; i++){
  if(objects[i] instanceof Edible)
    ---print(((Edible)objects[i]).howToEat());

  if(objects[i] instanceof Animal)
    ---print(((Animal)objects[i]).sound());
```

*) بعض الأحيان اكون ملزم بـ instanceof & Casting
← لأن ال methods بينهم مختلفة (وانا اريد استخدامها)
← احتاج لأتأكد من النوع

لو الاثنين نفس، وال method نفسها، استغني عن instanceof

Comparable Interface

```
public interface Comparable <E> {
    int compareTo(E e);
}
```

(E → Generic type)

← هذه abstract method ولا داعي لكتابة ذلك، لأنه دائمًا

Note:

① داخل ال interface اي متغير هو constant

② اي method هي abstract

③ *) دائمًا كل الصفات وال interface دائمًا كلها Public حتى لو لم يكتب

ex
```
public interface X {
    int a=8;  ──────→ constant
    void method(); ──────→ abs. method
}
```

Note ← لا مانع ان اكتب انا abstract او final
لكن لا مشكلة بعدم الكتابة

Note. ⊛ معنى interface كونها abstract ، الابن لازم يطبق كل abs. method
والا بكونه كلاسه abs. مثلها

Note. Even though I can't create an objet which its actual type is an abstract one. But I can use the abstract type in instance of or to create array.

ex. if (__ instance of Shape) ✓
Shape [] shapes = __ ✓
Shape [] shapes = { new Shape() } ✗ (not actual one)

Note String class is comparable.

---

ex String [] names = {"Ali", "Zaid", "Rami"};
Arrays.sort(names);
انتبه ArrayList في ⇐ Collections.sort(__)
or ___print("Ali".compareTo("Rami"));
يرتب حسب ASCII تصاعديا ، فالأول كلمة Ali ثم .....

الآن اريد استخدام هذه الinterface في كلاسات انا اعملها لكي يقارن بينهم
(Shape, Triangle) كيف ؟ على مثال

اكل: يلزمني أن اوضح له على أي أسس سوف يقارنه.

So: ex
Public abstract ~~class~~ class Shape implements Comparable<Shape>{
string color;
__
__

(override) يلزم أنه اعمل public int compareTo(Shape s) {
implementation
if (getArea() > s.getArea()) return 1;
else if (getArea() < s.getArea()) return -1;
أخذها من الكلاس

لو بدي ارتب تنازلي S.

```
if (getArea() < s.getArea())
    return 1;
```
واكمل...

Note: It's not allowed that the super class is Comparable and the subclass is comparable.
Must be one of them.

*) اذا عملت implements Comp. للاب ← يمتنع أن اعملها لاحد الابناء
*) يجوز ان اعملها في اكثر من ابن طالما لم اعمل للأب.
لكن لا استطيع انه اقارنه عند نوع الاب حتى لو النوع الفعلي (actual) هو عند نوع الابن

ex. suppose I let the triangle comparable.

```
Shape [] shapes = {new Triangle(), new Triangle()};
Arrays.sort(shapes); ✗
```
لانه ظاهريا هو shape ← not comparable
← قبل ان يعمل run ← الخطأ في Build

// إلا بحالة

```
Shape s = new Tringle();
Triangle t = new Tringle();

Triangle [] triangles = {t, ((Triangle)s)}
```

*) لنفترض أني استخدمت instance of عشان اتأكد

// ~~clone~~ interface قبل clone نبذة عن

How to clone (copy) an object ?

Remember, Immutable classes must ~~copy~~ return a copy of mutable types.

ex. Date d1 = new Date (10, 10, 20);

انتبه ، clone تغير من نوع object لذلك يلزمك type casting

Date d2 = (Date) d1.clone();

--- print(d1.equals(d2));
--- print (d1 == d2);

output: 1/ true 2/ false

Before, to copy an array, We used System.arraycopy.

// other way to do that.

ex. int [] arr1 = {1, 2, 3}
int [] arr2 = (int[]) arr1.clone();

Note We can clone on ArrayList, it is cloneable

Note String class is not cloneable (it is Imutable لا يصلح clone له لانه لا يتحول)

StringBuilder is not cloneable

Q if i'm asked to clone a string to another string:

Way 1

```
String x = "Hi";
Char [] arr1 = x.toCharArray();
Char [] arr2 = (char[]) arr1.clone();
x.
```
ثم (؟)

Way 2

```
String x = "Hi";
String y = x + "";
-- print (x.equals(y));
-- print (x == y);
```

output

true
false

عموماً ← لا نستخدم clone مع string

لانها Not Cloneable

ملاحظات مهمة عن لاب 8

1) اذا كنت رح استعمل instance of مع else if

دائماً احط الابناء بالاول

ex ~~s instanceof Worker~~

```
if (s instance of Worker) --
else if (s instance of Person) --
```

حتى يفحص الابن اولاً، لاني اذا حطيت الاب بالاول

هو دائماً instanceof Person ← لا يدخل الابن

2) استعمل else if وليس if

حتى لا يدخل بالثنين

3) اذا كان الاب abstract وله abs. method
وكان الابن Concrete ⇐ تلقائياً يجب ان تنفذ
كل abs. method الي في الاب حتى ولم يذكرها في UML
اوحتى لو لم يطلبها

// Cloneable Interface

→ public interface Cloneable { }
// marker interface ← فارغة، فقط هيك
~~clone object في موجودة~~

// Method Clone() is in object class, Not in the interface
كيف يكون تعريفها داخل Object class ؟

→ protected native Object clone()
throws CloneNotSupportedException

*) هي protected لكن عندما اعمل لها override ⇐ الأفضل أن احولها Public
*) اذا كان الـ permission في الأب محدوداً ⇐ يمكن ان اخفف القيود في الابن
*) لا يمكن ان اضيّق الصلاحيات في الابن، اكثر من التي في الأب.

How to Clone Person ?

```
public class Person implements Cloneable {
    String name; int id; Date birthDate;

    public Object clone() throws CloneNotSupportedException {   // override   (1)
        return super.clone();
    }
```

But I must handle the exception

→ A) public static void main() throws CloneNotSupportedException
هذه طريقة بسيطة يعرض لي هذا الـ Exception إن حصل

→ (1) is Surface Cloning // shallow cloning

B) الطريقة الأصح لعمل handling

```
try {
    Person P1 = (Person) P.clone();
}
catch (CloneNotSupportedException) {
    ___ print("Do not worry");
}
```

هكذا اذا لم يحصل error ← لا يدخل Catch
← اذا حصل ، يطبع don't worry ويكمل البرنامج

Now, ___ Println (P.name == P1.name);

output → true! Why?
This is cloning; the cloned object will have same values at the original same address.

تعريف (P = P1 // يصبح كأنه يؤشر تأشير (مثل
* لو عدلت عالأول ، يعدل على الثاني
ما الحل؟

(2) Deep Cloning.

أحمل الـ override هكذا

```
Public Object clone() thrown CloneNotSupportedException {
    Person pCopy = (Person) super.clone();        // ننسخ اول level
    pCopy.birthDate = (Date) birthDate.clone();   // }
    pCopy.name = name + "";                       // } ننسخ المستوى الثاني
    return pCopy;
}
```

← بعد ان ننسخنا أول level .
← ابدأ بنسخ كل reference داخلي ، واحدة تلو الأخرى .
← لا أهتم لـ int , Char , double , ...
لأنهم ليس reference .

// An interface inherites from interfaces :

Public interface a extends b, c, d {

// But Class inherites from interfaces {

Public class X implements a {

Class X must implement all abstract method in interfaces a, b, c, d.
Other Wise, X must be abstract.

= class extends class
class implements interface
interface extends interface

Note. Chapter of Exception & files is not in this Notes.
