Primitive types values as objects

Wrapper classes: (capital)

Integer, Float, Double, Boolean, Character, Long, Byte, Short

ex
```
Integer x = new Integer(4);
Integr y= new Integer("4");
Integr 2 = 4;   auto boxing
```
} Objects تخزن

```
... Println(y.doubleValue());
```
→ 4.0 Simple type

```
... Print(y.intValue());
```
→ 4 Simple type

طريقتين مختلفين، لانه Static

```
... Print(X.parseInt("234");
... Print(Integer.parseInt("234");
```

simple type إلى String من تحول

→ 234
→ 234

{ ⊛ X.parseInt (or Double or ...)
simple type تأخذ لتحول لـ string

Note, parseInt takes radix, too.

ex. print(Integer.parseInt("10", 8));
→ 8 simple

---

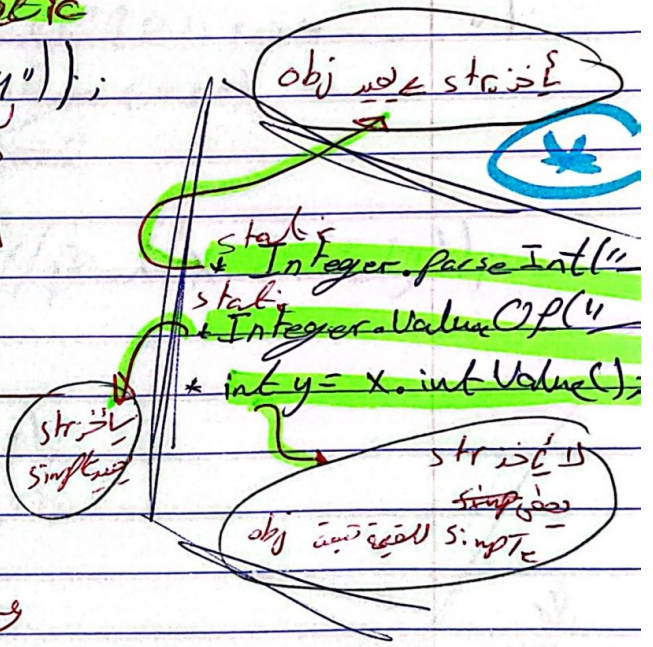
parseInt و valueOf الفرق بين

parseInt → Simple type تعيد

valueOf → object int (wrapper type)

print(x.valueOf("34"));
static
هنا تعيد object

Simple لكن صنعه وتعامل معه ↓
auto unBoxing على



---

ex Integer x = 9;
auto Boxing

Boxing ~~auto~~

ex Integer[] arr = { new Integer(4), new Integer(3), 5 };

print(arr[0] + arr[1]);

هنا object كيف جمع؟ لازم unBoxing ثم جمع

وتخزن الاجابة simple

---

arr[1] = arr[0] + 1;
unBoxing
Boxing

كلاهما ليس (wrapper)

// BigInteger , ~~BigDe~~ BigDecimal

ارقام غير محدودة الخانات

(*) ضروري
```
import java.math.*;
```

(*) لازم يأخذ String

(*) العمليات الحسابية تكون من خلال methods

ex
```
BigDecimal x = new BigDecimal("9011234___");
BigInteger y = new ~~BigDecimal~~ BigInteger("34");
__print(x.multiply(y)); //print(x.divide(y));
```

ex
```
____ x = ____ ("5");
____ y = ____ ("3");

print(x.divide(y));
```
error (غير منتهي)

الحل للمشكلة
```
print(x.divide(y, 10));
```
لعشرة خانات

او
```
print(x.divide(y, 10, BigDecimal.Round_UP));
```
لعشر خانات يقرب للاعلى

يعمل على BigInteger على عكس BigDecimal نفس العمليات فقط

Q write a method, calculates fact as BigInteger

```
Print (fact(5));

Public static BigInteger fact (int n){

    BigInteger result = new BigInteger("1");
                      = 1 ;   ممنوع

            Wrapping لا يعمل ⇐

    for (int i=1 ; i<= n ; i++)
        result = result. multiply(new BigInteger (i+""));

    return result;
}
```

ممنوع اعطي i
اعطي "" i+ حتى يأخذها string

CMD اعرف هو شو

```
class Compute {
  Public Static Void main(String [] args){      Array
    int result=0;
    ___ Println( args. length);
    switch(args[1]. charAt(0)) {
  string, دخل ⇐ case '+' : result= Integer. parseInteger (args[0]) + ___ (args[2]
  ([1,0,2]
    }
  خرج  Print( result);
```

اذا دخلت 7 + 3

args[0] ↑ 7, args[1] ↑ +, args[2] ↑ 3

→ 3   char عدد
  11 المجموع

unBoxing يعمل هو parseInt

(*) in Strings, split method

⇒ use "[ ]" with regex

(*) in Strings, replaceAll method

⇒ use "\\ ####" with regex

Note

Double x = 40; ✗

Double x = 40.0; ✓

] لازم يكون obj دائما

ex

```
Double x = 40.1;
print(x.intValue());
```

⇒ 40