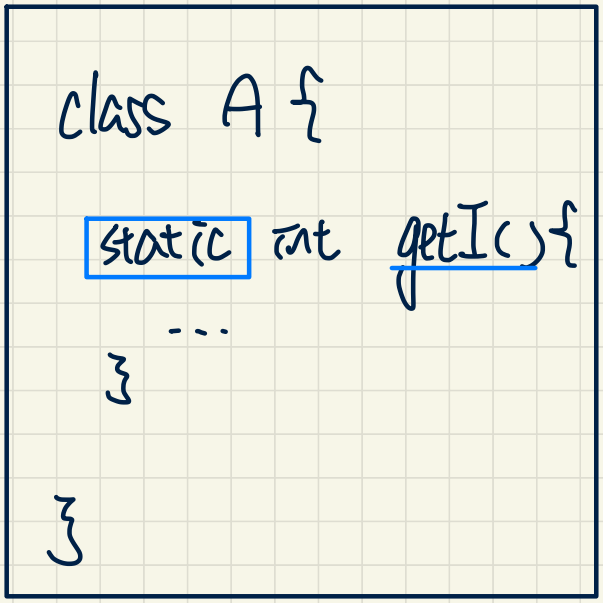
EECS1022 Programming for Mobile Computing (Winter 2021)

Q&A - Lectures

Monday, April 5



A.getI()

① A is a class name
(not really an object)

② A oa = new A();
oa.getI(); ✓

warning: invoking static method in a non-static way.

Modifier and Type	Field and Description
static double	**E** The double value that is closer than any other to *e*, the base of the natural logarithms.
static double	**PI** The double value that is closer than any other to *pi*, the ratio of the circumference of a circle to its diameter.

not a method

```
public class Math {
    public static final double PI = 3.14;
}
```

→ Math.PI ✓

```
public class Math2 {
    private static final double PI = 3.14;
    public static double getPI { ... }
}
```

→ Math.PI ✗

Math.getPI(). ✓

1 100

[0.0, 1)

static double	**random**()
	Returns a double value with a positive sign, greater than or equal to 0.0 and less than 1.0.

```
(int) Math.random() * 100
```

or

```
(int) (Math.random() * 100)
```

(int) 0.12 → 0 * 100 = 0

Math.random() → 0.0, 0.12, 0.99 ⋮

0.12 * 100 → 12.00 → 12.

String.format("%.2f", Math.random())

double d = 99.9;

int i = (int) 99.9;

truncate all digits after decimal point (no rounding).

i → 100? X

i → 99.

API: ArrayList<E>

generic parameter.
↳ instantiate it by the type of elements in the list.

int	**size**() Returns the number of elements in this list.
boolean	**add**(E e) Appends the specified element to the end of this list.
void	**add**(int index, E element) Inserts the specified element at the specified position in this list.
boolean	**contains**(Object o) Returns true if this list contains the specified element.
E	**remove**(int index) Removes the element at the specified position in this list.
boolean	**remove**(Object o) Removes the first occurrence of the specified element from this list, if it is present.
int	**indexOf**(Object o) Returns the index of the first occurrence of the specified element in this list, or -1 if this list does not contain the element.
E	**get**(int index) Returns the element at the specified position in this list.

String Point

String Point

E / String · Point

↳ I → removed
→ not removed

You can pass any type of object.

① l1.remove("alan") ✓
⑧ l1.remove(23) ✓
⑨ l1.remove(p1) ✓

ArrayList<String> l1 = ..
ArrayList<Point> l2 = ..

① l1.add("alan") ✓
② l1.add(23) X
③ l1.add(p1) X

④ l2.add("alan"); X
⑤ l2.add(23); X
⑥ l2.add(p1); ✓
↳ Point

ArrayList< ? > list = ...

1. Any library class
   e.g. String, ArrayList, Hashtable.

2. Any class you created
   e.g. Factory.

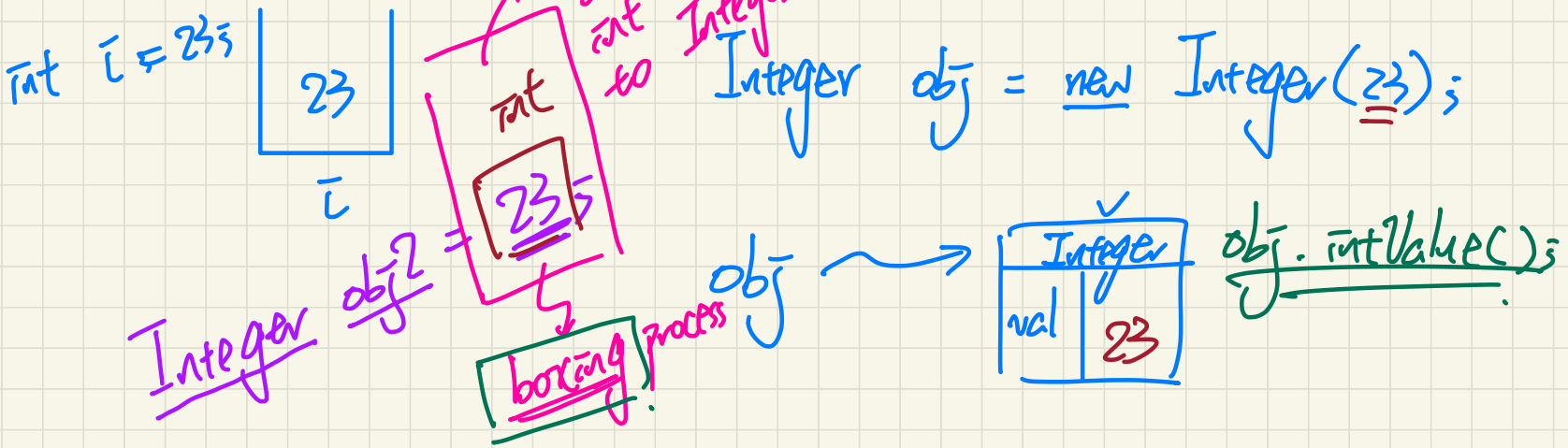
3. no primitive type e.g. ArrayList<~~int~~> list = ...
   Instead, use wrapper class e.g. ArrayList<Integer>

int → primitive type

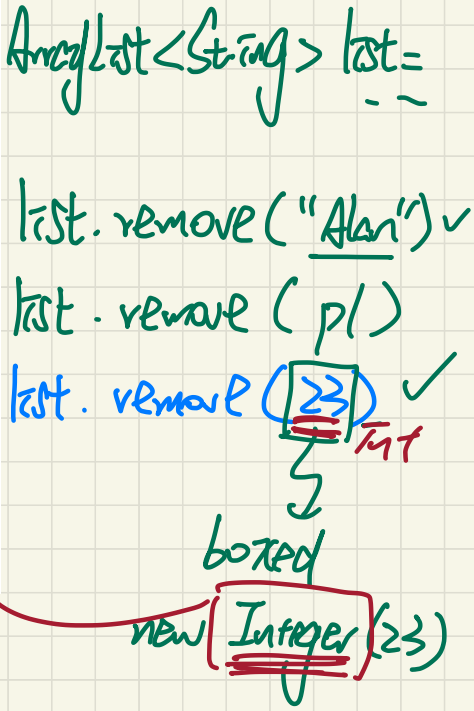
vs

Integer → reference type

Primitive	Wrapper
int	Integer
float	Float
double	Double
char	Character



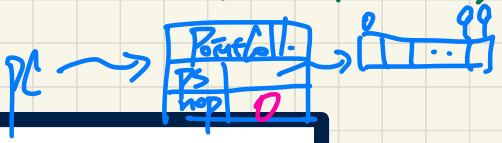
int	**size**() Returns the number of elements in this list.
boolean	**add**(E e) Appends the specified element to the end of this list.
void	**add**(int index, E element) Inserts the specified element at the specified position in this list.
boolean	**contains**(Object o) Returns `true` if this list contains the specified element.
E	**remove**(int index) Removes the element at the specified position in this list.
boolean	**remove**(Object o) Removes the first occurrence of the specified element from this list, if it is present.
int	**indexOf**(Object o) Returns the index of the first occurrence of the specified element in this list, or -1 if this list does not contain the element.
E	**get**(int index) Returns the element at the specified position in this list.



Use of ArrayList<String>

```
class PointCollector {
  Point[] ps;
  int nop;
}
```

```
PointCollector pc =
  new PointCollector();
```




```
1  import java.util.ArrayList;
2  public class ArrayListTester {
3    public static void main(String[] args) {
4      ArrayList<String> list = new ArrayList<String>();
5      println(list.size());
6      println(list.contains("A"));
7      println(list.indexOf("A"));
8      list.add("A");
9      list.add("B");
10     println(list.contains("A")); println(list.contains("B")); println(list.contains("C"));
11     println(list.indexOf("A")); println(list.indexOf("B")); println(list.indexOf("C"));
12     list.add(1, "C");
13     println(list.contains("A")); println(list.contains("B")); println(list.contains("C"));
14     println(list.indexOf("A")); println(list.indexOf("B")); println(list.indexOf("C"));
15     list.remove("C");
16     println(list.contains("A")); println(list.contains("B")); println(list.contains("C"));
17     println(list.indexOf("A")); println(list.indexOf("B")); println(list.indexOf("C"));
18
19     for(int i = 0; i < list.size(); i ++) {
20       println(list.get(i));
21     }
22   }
23 }
```

≈ pc.nop

capacity ≈ MAX_NUMBER_OF_POINTS.