A review of some Java basics

Java pass-by-value and List<> references

Java is always *pass-by-value*

- "Java is always **pass-by-value**. Unfortunately, they decided to call the location of an object a *reference*. When we pass the value of an object, we are passing the *reference* to it. This is confusing to beginners."

http://stackoverflow.com/questions/40480/is-java-pass-by-reference-or-pass-by-value

Example 1

```
public static void main( String[] args ) {
    Dog aDog = new Dog("Max");
    foo(aDog);
    if (aDog.getName().equals("Max")) {
        // ???
    }
    if (aDog.getName().equals("Fifi")) {
        // ???
    }
}
public static void foo(Dog d) {
    if (d.getName().equals("Max")){ // ??? }
    d = new Dog("Fifi");
    if (d.getName().equals("Fifi")) { // ??? }
}
```

Example 1 - solution

```
public static void main( String[] args ) {
    Dog aDog = new Dog("Max");
    foo(aDog);
    if (aDog.getName().equals("Max")) {
        // true, java passes by value
    }
    if (aDog.getName().equals("Fifi")) {
        // false
    }
}
public static void foo(Dog d) {
    if (d.getName().equals("Max")) { // true }
    d = new Dog("Fifi");
    if (d.getName().equals("Fifi")) { // true }
}
```

Example 2

```
public static void main( String[] args ) {
    Dog aDog = new Dog("Max");
    foo(aDog);
    if (aDog.getName().equals(”Max")) {
        // ???
    }
    if (aDog.getName().equals("Fifi")) {
        // ???
    }
}
public static void foo(Dog d) {
    if (d.getName().equals("Max")) { // ??? }
    d.setName("Fifi");
    if (d.getName().equals("Fifi")) { // ??? }
}
```

Example 2 - solution

```
public static void main( String[] args ) {
    Dog aDog = new Dog("Max");
    foo(aDog);
    if (aDog.getName().equals(”Max")) {
       // false
    }
    if (aDog.getName().equals("Fifi")) {
       // true, java passes by value
    }
}
public static void foo(Dog d) {
    if (d.getName().equals("Max")) { // true }
    d.setName("Fifi");
    if (d.getName().equals("Fifi")) { // true }
}
```

Java List<> stores references

- The *List* is an ordered collection that contains object references

- If I place an object in a list and later on change one of its values, is the one in the list going to have the updated value?
  - It is possible to change/update only *mutable* objects. It is not possible to update *immutable* (e.g List<String>) objects via their references
  - Immutable means that once the constructor for an object has completed execution that instance can't be altered.

Example

```
public static void main( String[] args ) {
    List<Pos> list = new LinkedList<Pos>();
    Pos a = new Pos(2,3);
    Pos b = new Pos(3,4);
    list.add(a);
    list.add(b);

    for (Pos p : list)
       System.out.println(p)

    a.setX(11);

    for (Pos p : list)
       System.out.println(p)
}
```

Example

```
public static void main( String[] args ) {
    List<Pos> list = new LinkedList<Pos>();
    Pos a = new Pos(2,3);
    Pos b = new Pos(3,4);
    list.add(a);
    list.add(b);

    for (Pos p : list)                  (2,3)
       System.out.println(p)            (3,4)

    a.setX(11);

    for (Pos p : list)                  (11,3)
       System.out.println(p)            (3,4)
}
```

Case 1 (immutable objects)

```
public static void main( String[] args ) {
    String foo = "foo";
    String bar = "bar";
    ArrayList<String> list = new ArrayList<String>();
    list.add(foo);
    list.add(bar);
    foo = "foo2";
    System.out.println(list.get(0));
}
```

Case 1 (immutable objects)

```
public static void main( String[] args ) {
    String foo = "foo";
    String bar = "bar";
    ArrayList<String> list = new ArrayList<String>();
    list.add(foo);
    list.add(bar);
    foo = "foo2";
    System.out.println(list.get(0));
}
```

Output: foo

String are immutable objects in Java

Case 2 (autoboxing)

```
public static void main( String[] args ) {
    Integer i = 6;
    ArrayList<Integer> list = new ArrayList<Integer>();
    list.add(i);
    i = 8;
    System.out.println(list.get(0));
}
```

Case 2 (autoboxing)

```
public static void main( String[] args ) {
    Integer i = 6;
    ArrayList<Integer> list = new ArrayList<Integer>();
    list.add(i);
    i = 8;
    System.out.println(list.get(0));
}
```

Output: 6

At compile time:
i = new Integer(8)

POLITECNICO DI TORINO

e-Lite

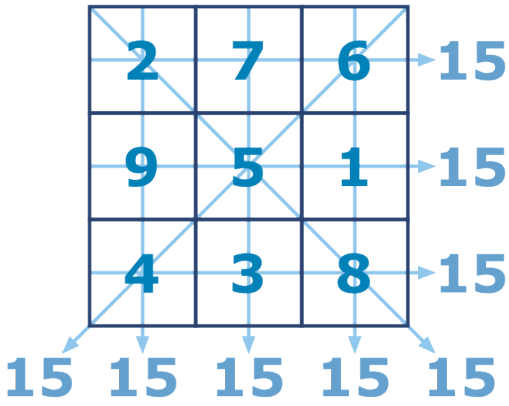
Recursion Excercise – Magic Square

Tecniche di Programmazione – A.A. 2016/2017

Magic Square

- A magic square is $n * n$ grid (where n is the number of cells on each side) filled with distinct positive integers in the range $1, 2, \ldots, n^2$ such that each cell contains a different integer and the sum of the integers in each row, diagonal and column is equal

- The sum is called *magic constant*.



Magic Square

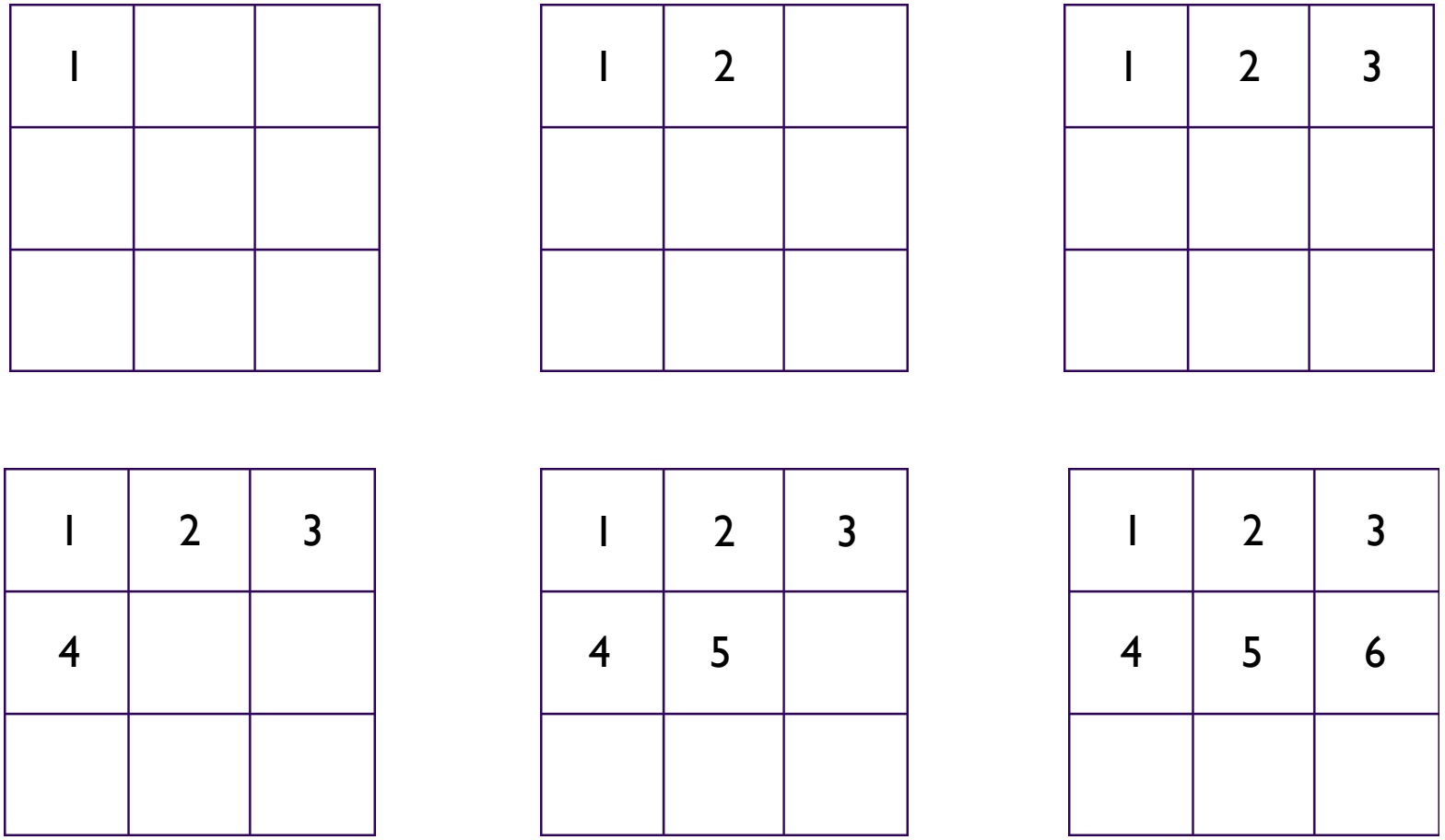
- *magic constant* $M = \frac{n(n^2+1)}{2}$
  - For normal magic squares of orders *n* = 3, 4, 5, 6, 7, and 8, the magic constants are, respectively: 15, 34, 65, 111, 175, and 260

- *There exist exact methods for constructing magic squares of even or odd order…*
- *… while we will use recursion*

Design tips

Recursion

Analizzare il problema



Analizzare il problema

1	2	3
4	5	6
7		

1	2	3
4	5	6
7	8	

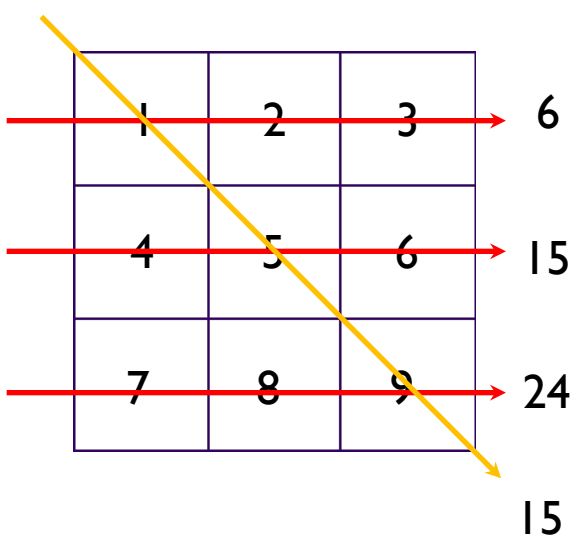
1	2	3
4	5	6
7	8	9

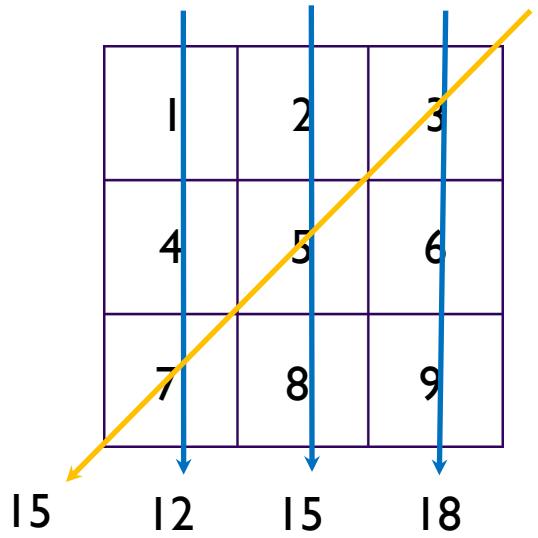
Analizzare il problema

1	2	3
4	5	6
7		

1	2	3
4	5	6
7	8	

1	2	3
4	5	6
7	8	9



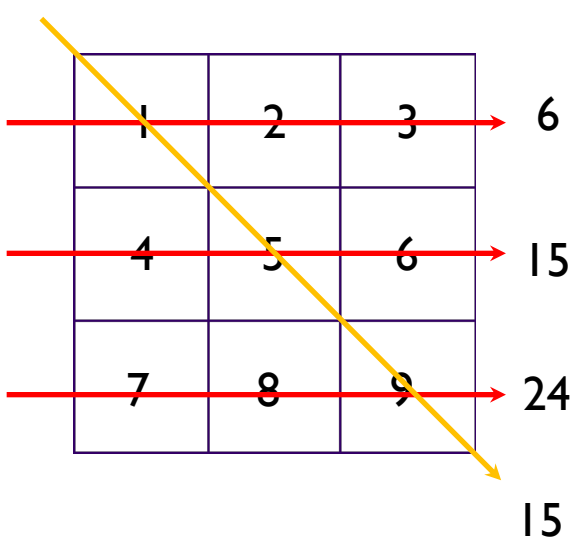


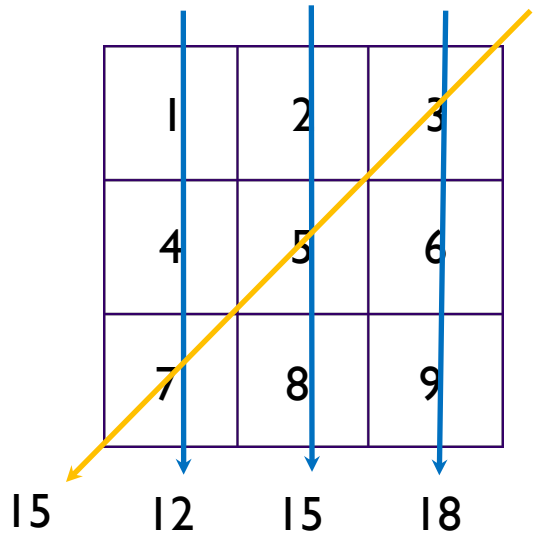
Analizzare il problema

1	2	3
4	5	6
7		

1	2	3
4	5	6
7	8	

1	2	3
4	5	6
7	8	9





Analizzare il problema

- Come imposto in generale la ricorsione?
- Che cosa mi rappresenta il "livello"?
- Qual è il livello massimo?
- Com'è fatta una soluzione parziale?
- Com'è fatta una soluzione completa?
- Come viene avviata la ricorsione (livello 0)?
- Come genero Parziale(i+1) partendo da Parziale(i)?

Analizzare il problema

- Come imposto in generale la ricorsione?
  - Ad ogni passo inserisco un numero $1, 2, \ldots, n^2$ nella prima casella libera del quadrato. Delego al passo successivo il riempimento delle successive caselle

Analizzare il problema

- Come imposto in generale la ricorsione?
  - Ad ogni passo inserisco un numero $1, 2, \ldots, n^2$ nella prima casella libera del quadrato. Delego al passo successivo il riempimento delle successive caselle
- Che cosa mi rappresenta il "livello" i-esimo?
  - Il livello rappresenta la casella i-esima del quadrato che devo riempire

Analizzare il problema

- Come imposto in generale la ricorsione?
  - Ad ogni passo inserisco un numero $1, 2, \ldots, n^2$ nella prima casella libera del quadrato. Delego al passo successivo il riempimento delle successive caselle
- Che cosa mi rappresenta il "livello" i-esimo?
  - Il livello rappresenta la casella i-esima del quadrato che devo riempire
- Qual è il livello massimo?
  - n^2, l'ultima casella

Analizzare il problema

- Com'è fatta una soluzione parziale i-esima?
  - Quadrato riempito fino alla casella i-esima

1	2	3
4	5	

Analizzare il problema

- Com'è fatta una soluzione parziale i-esima?
  - Quadrato riempito fino alla casella i-esima

1	2	3
4	5	

- Com'è fatta una soluzione completa?
  - Il quadrato completo di tutti numeri $1, 2, \ldots, n^2$

1	2	3
4	5	6
7	8	9

Analizzare il problema

- Come viene avviata la ricorsione (livello 0)?
  - La ricorsione inizia al livello 0 con il quadrato vuoto

Analizzare il problema

- Come viene avviata la ricorsione (livello 0)?
  - La ricorsione inizia al livello 0 con il quadrato vuoto
- Come genero Parziale(i+1) partendo da Parziale(i)?

```
for (i = 0; i< n*n; i++)
    if (!parziale.contains(i))
        parziale.add(i)
```

Identificare le soluzioni valide

- Data una soluzione **parziale**, come sapere se è valida?
  - Non esistono soluzioni parziali valide

Identificare le soluzioni valide

- Data una soluzione **parziale**, come sapere se è valida?
  - Non esistono soluzioni parziali valide
- Data una soluzione **completa**, come sapere se è valida?
  - Calcolo la somma del numero di righe, di colonne e delle diagonali. Confronto con magic number

Identificare le soluzioni valide

- Data una soluzione **parziale**, come sapere se è valida?
  - Non esistono soluzioni parziali valide
- Data una soluzione **completa**, come sapere se è valida?
  - Calcolo la somma del numero di righe, di colonne e delle diagonali. Confronto con magic number
- Cosa devo fare con le soluzioni complete valide?
  - Fermarmi alla prima? ok
  - Generarle e memorizzarle tutte? ok
  - Contarle? ok

Progettare le strutture dati

- Qual è la struttura dati per memorizzare una soluzione (parziale o completa)?
  - ArrayList<Integer>, int[], int[][]

Progettare le strutture dati

- Qual è la struttura dati per memorizzare una soluzione (parziale o completa)?
  - ArrayList<Integer>, int[], int[][]
- Qual è la struttura dati per memorizzare lo stato della ricerca (della ricorsione)?
  - Variabile intera step.

Scheletro del codice

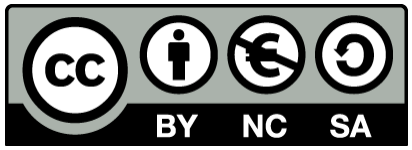
```
// Struttura di un algoritmo ricorsivo generico

void recursive (... , level) {

  // E -- sequenza di istruzioni che vengono eseguite sempre
  // Da usare solo in casi rari (es. Ruzzle)
  doAlways();

  // A
  if (condizione di terminazione) {
    doSomething;
    return;
  }

  // Potrebbe essere anche un while ()
  for () {

    // B
    generaNuovaSoluzioneParziale;


    if (filtro) { // C
      recursive (..., level + 1);
    }

    // D
    backtracking;
  }

}
```

Riempire lo scheletro (del codice)

Blocco	Frammento di codice
A	
B	
C	
D	
E	

```
// Struttura di un algoritmo ricorsivo

void recursive (... , level) {

  // E -- sequenza di istruzioni che ve
  // Da usare solo in casi rari (es. Ru
  doAlways();

  // A
  if (condizione di terminazione) {
    doSomething;
    return;
  }

  // Potrebbe essere anche un while ()
  for () {

    // B
    generaNuovaSoluzioneParziale;


    if (filtro) { // C
      recursive (..., level + 1);
    }

    // D
    backtracking;
  }

}
```

Licenza d'uso



- Queste diapositive sono distribuite con licenza Creative Commons "Attribuzione - Non commerciale - Condividi allo stesso modo (CC BY-NC-SA)"
- Sei libero:
  - di riprodurre, distribuire, comunicare al pubblico, esporre in pubblico, rappresentare, eseguire e recitare quest'opera
  - di modificare quest'opera

- Alle seguenti condizioni:
  - **Attribuzione** — Devi attribuire la paternità dell'opera agli autori originali e in modo tale da non suggerire che essi avallino te o il modo i cui tu usi l'opera.

  - **Non commerciale** — Non puoi usare quest'opera per fini commerciali.

  - **Condividi allo stesso modo** — Se alteri o trasformi quest'opera, o se la usi per crearne un'altra, puoi distribuire l'opera risultante solo con un licenza identica o equivalente a questa.

- http://creativecommons.org/licenses/by-nc-sa/3.0/