

Programmazione I-B 2020-21

Esercitazioni

Attilio Fiandrotti

attilio.fiandrotti@unito.it

28 Gennaio 2021

Esercitazioni – Programma

- Mer 23 Dic 11:00-13:00
- Gio 7 Gen 9:00-11:00
- Gio 14 Gen 14:00-16:00 (Baroglio)
- Gio 21 Gen 9:00-11:00 (Beccuti)
- ~~Mar 26 Gen 9:00-11:00~~ -> da riallocare entro Gennaio
- Gio 28 Gen 14:00-16:00

Orari del corso

- se (matricola pari) allora T2
- se (matricola dispari) allora T1

Roversi

Fiaudrotti

Roversi

Orario delle lezioni del primo anno del Corso B

Ora	Lun	Mar	Mer	Gio	Ven	Sab
9-10	Prog I B (Aula B)	Log B (Aula B)	Mate Discr B (Aula B)	Prog I B T2 (Laboratorio Turing)	Mate Discr B (Aula B)	
10-11	Prog I B (Aula B)	Log B (Aula B)	Mate Discr B (Aula B)	Prog I B T2 (Laboratorio Turing)	Mate Discr B (Aula B)	
11-12	Log B (Aula B)	Mate Discr B (Aula B)	Prog I B (Aula B)	Prog I B T2 (Laboratorio Turing)		
12-13	Log B (Aula B)	Mate Discr B (Aula B)	Prog I B (Aula B)			
13-14				Prog I B T1 (Laboratorio Turing)		
14-15	RO B (Aula B)	RO B (Aula B)	Ingl I (Aula A)	Prog I B T1 (Laboratorio Turing)		
15-16	RO B (Aula B)	RO B (Aula B)	Ingl I (Aula A)	Prog I B T1 (Laboratorio Turing)		
16-17						
17-18						
18-19						

Correzione slides

Slides aggiornate

- Aggiornate le slides dell'esercitazione del 7 Gennaio per coerenza con le convenzioni adottate nelle dispense

Esempio: calcolo della circonferenza

In Java i parametri sono posizionali

```
float circ = circonferenza(pi, raggio);
```

```
public static float circonferenza(float pi, float raggio) {  
    float res = 2 * pi * raggio;  
    return res;  
}
```



```
public static float circonferenza(float topolino, float pippo) {  
    float res = 2 * topolino * pippo;  
    return res;  
}
```

Il modello di memoria JVM (semplificato)

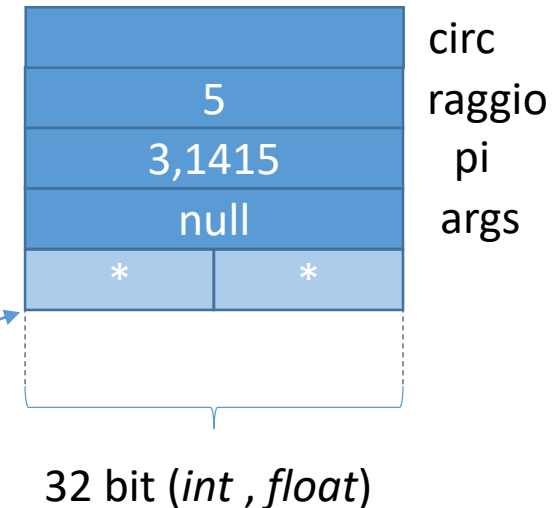
Prossima linea da eseguire (non ancora eseguita)

Indirizzi crescenti

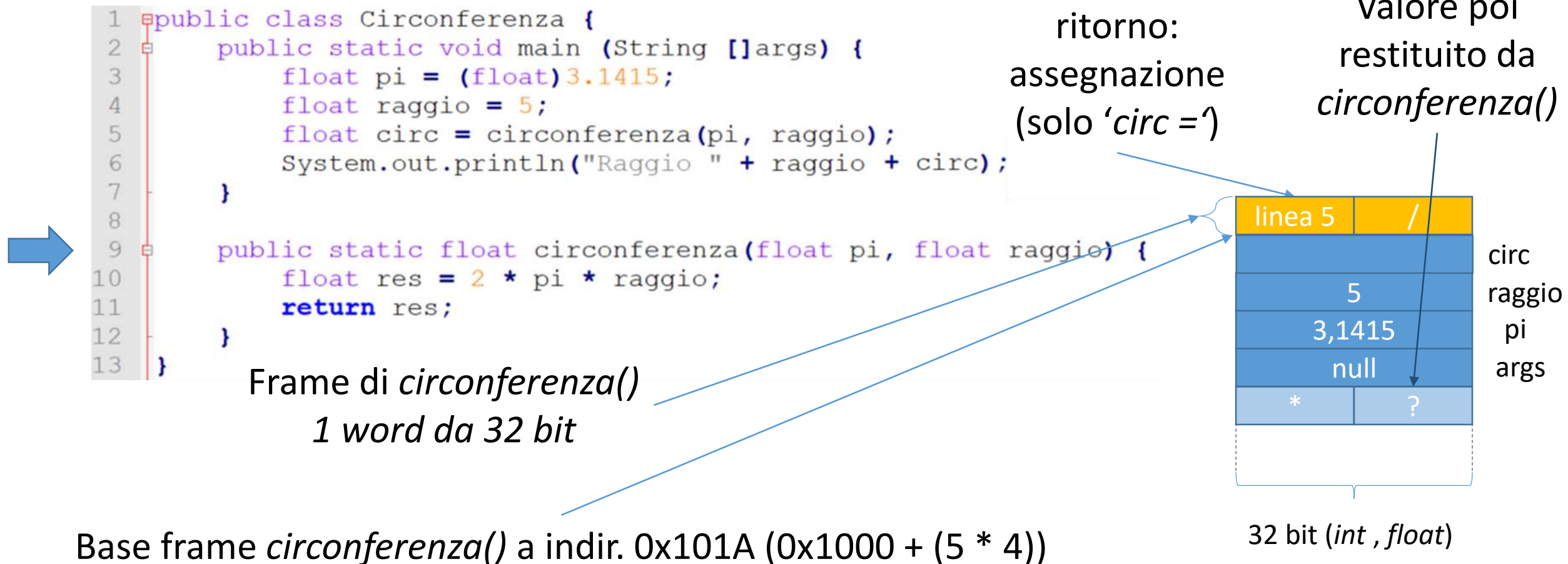
```
1 public class Circonferenza {
2     public static void main (String []args) {
3         float pi = (float)3.1415;
4         float raggio = 5;
5         float circ = circonferenza(pi, raggio);
6         System.out.println("Raggio " + raggio + circ);
7     }
8
9     public static float circonferenza(float pi, float raggio) {
10         float res = 2 * pi * raggio;
11         return res;
12     }
13 }
```

Frame di *main()*
5 words da 32 bit

Base del frame , es: 0x1000



Il modello di memoria JVM (semplificato)

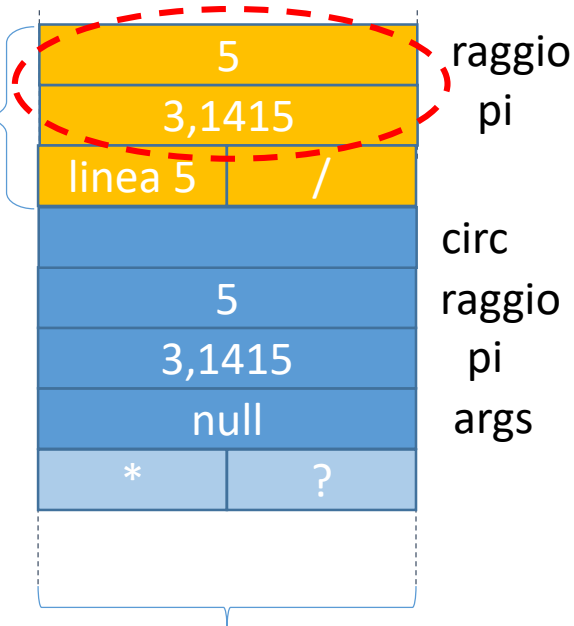


Il modello di memoria JVM (semplificato)

Parametri di *circonferenza()*
pi e *raggio* nello stack

```
1 public class Circonferenza {
2     public static void main (String []args) {
3         float pi = (float)3.1415;
4         float raggio = 5;
5         float circ = circonferenza(pi, raggio);
6         System.out.println("Raggio " + raggio + circ);
7     }
8
9     public static float circonferenza(float pi, float raggio) {
10         float res = 2 * pi * raggio;
11         return res;
12     }
13 }
```

Frame di *circonferenza()*
3 words da 32 bit



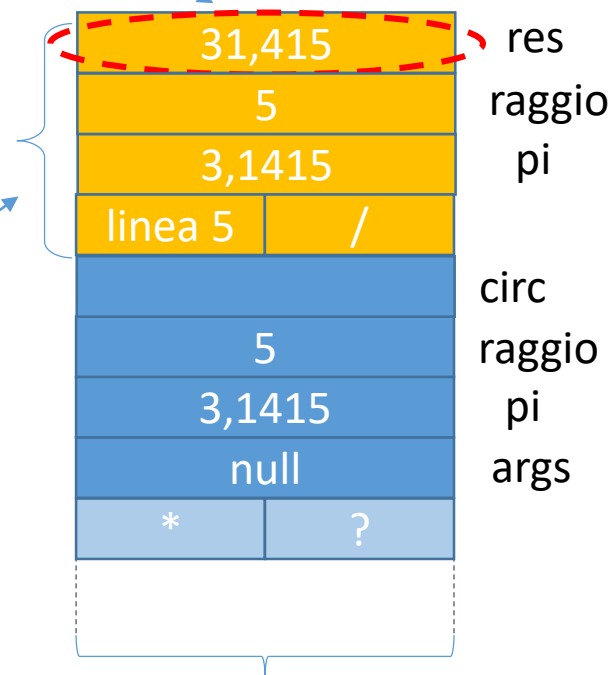
32 bit (int , float)

Il modello di memoria JVM (semplificato)

res per ora nel frame
di *circonferenza()*

```
1 public class Circonferenza {  
2     public static void main (String []args) {  
3         float pi = (float)3.1415;  
4         float raggio = 5;  
5         float circ = circonferenza(pi, raggio);  
6         System.out.println("Raggio " + raggio + circ);  
7     }  
8  
9     public static float circonferenza(float pi, float raggio) {  
10        float res = 2 * pi * raggio;  
11        return res;  
12    }  
13 }
```

Frame di *circonferenza()*
4 words da 32 bit

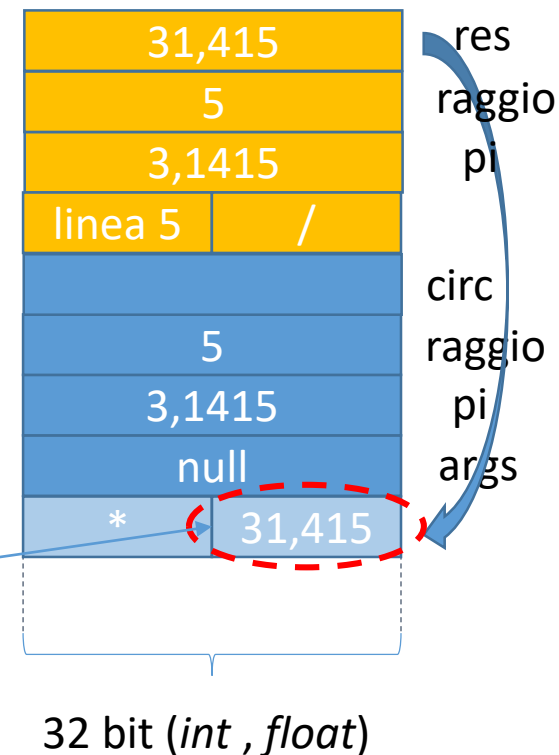


32 bit (int , float)

Il modello di memoria JVM (semplificato)

```
1 public class Circonferenza {
2     public static void main (String []args) {
3         float pi = (float)3.1415;
4         float raggio = 5;
5         float circ = circonferenza(pi, raggio);
6         System.out.println("Raggio " + raggio + circ);
7     }
8
9     public static float circonferenza(float pi, float raggio) {
10         float res = 2 * pi * raggio;
11         return res;
12     }
13 }
```

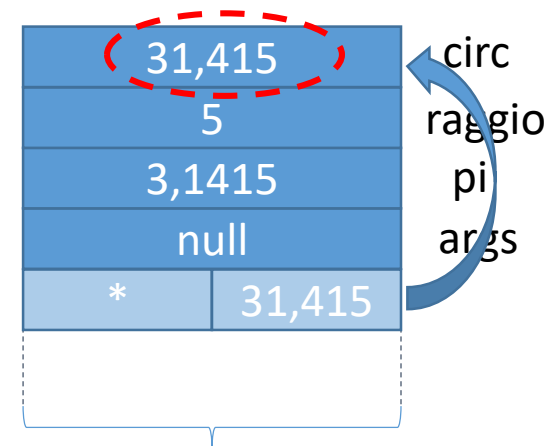
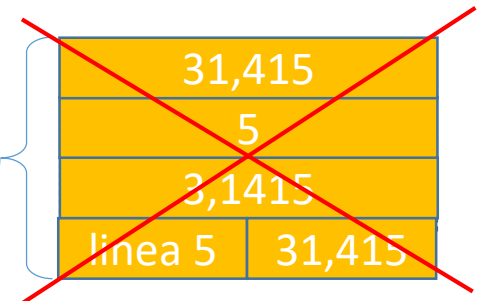
risultato di
circonferenza()
accessibile da
main()



Il modello di memoria JVM (semplificato)

```
1 public class Circonferenza {
2     public static void main (String []args) {
3         float pi = (float)3.1415;
4         float raggio = 5;
5         float circ = circonferenza(pi, raggio);
6         System.out.println("Raggio " + raggio + circ);
7     }
8
9     public static float circonferenza(float pi, float raggio) {
10         float res = 2 * pi * raggio;
11         return res;
12     }
13 }
```

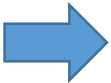
Frame di *circonferenza()*
eliminato
(memoria di nuovo
disponibile)



32 bit (int , float)

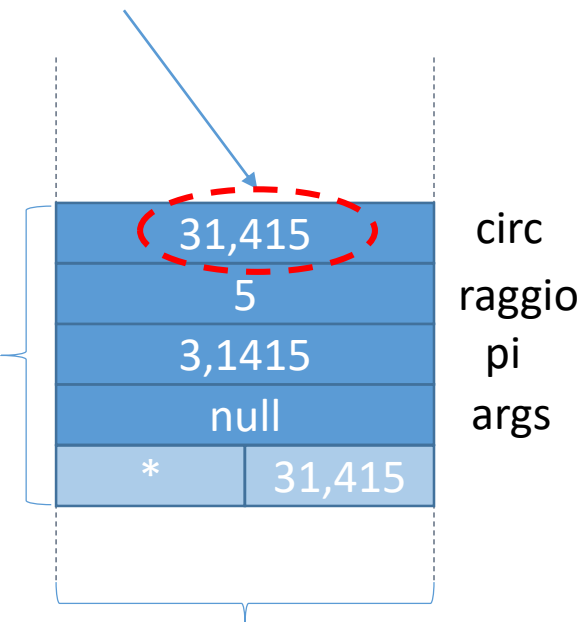
Il modello di memoria JVM (semplificato)

Risultato di *circonferenza()*
in variabile local *circ*
nel frame di *main()*



```
1 public class Circonferenza {
2     public static void main (String []args) {
3         float pi = (float)3.1415;
4         float raggio = 5;
5         float circ = circonferenza(pi, raggio);
6         System.out.println("Raggio " + raggio + circ);
7     }
8
9     public static float circonferenza(float pi, float raggio) {
10         float res = 2 * pi * raggio;
11         return res;
12     }
13 }
```

Frame di *main ()*



32 bit (*int* , *float*)

Esercizio: Torneo di calcio

Caso base –metodo *main()*

Torneo di calcio

- Si sviluppi un programma Java che, data una serie di partite di un torneo di calcio, i) stabilisca la capolista e ii) produca la classifica finale
- Esempio di partite di torneo triangolare (input del programma):

Torino-Roma 1-1
Roma-Napoli 0-2
Napoli-Torino 2-3

- Classifica risultante (output del programma):

Squadra	Punti
Torino	3
Napoli	2
Roma	1



Torneo di calcio - difficoltà

- La difficoltà maggiore consiste nel «tradurre» l'esercizio dalla forma testuale a quella algoritmica
- Due problemi fondamentali
 1. Quali strutture dati occorreranno
 2. Quali metodi serviranno
- E' fortemente suggerito risolvere i due problemi sopra in tale ordine

Le strutture dati

- Per ogni partita, sarà necessario memorizzare il nome delle due squadre che l'hanno disputata e il numero di goals segnati da ognuna
- Per la classifica sarà necessario memorizzare nome e punteggio per ogni squadra (2 punti vittoria, 1 pareggio, 0 sconfitta)
- Soluzione ideale: array di *strutture dati eterogenee* che al momento non sappiamo gestire (vedi *classi* in ProgrammazioneII)
- Come possiamo gestire la base dati utilizzando solo array di tipo omogeneo ?

Le strutture dati – le partite

- Utilizzeremo due array di String e due array di int (4 array in totale)

```
* L'esempio sottostante si legge come:  
* Torino-Roma 1-1  
* Roma-Napoli 0-2  
* Napoli-Torino 2-3  
*/  
String[] nomeSquadraCasa = {"Torino", "Roma", "Napoli"};  
String[] nomeSquadraOspite = {"Roma", "Napoli", "Torino"};  
int[] goalsSquadraCasa = {1, 0, 2};  
int[] goalsSquadraOspite = {1, 2, 3};
```

Le strutture dati – le partite

- Utilizzeremo due array di String e due array di int (4 array in totale)

```
* L'esempio sottostante si legge come:  
* Torino-Roma 1-1  
* Roma-Napoli 0-2  
* Napoli-Torino 2-3  
*/  
String[] nomeSquadraCasa = {"Torino", "Roma", "Napoli"};  
String[] nomeSquadraOspite = {"Roma", "Napoli", "Torino"};  
int[] goalsSquadraCasa = {1, 0, 2};  
int[] goalsSquadraOspite = {1, 2, 3};
```

- Esempio

```
println(nomeSquadraCasa[0] + " vs. " + nomeSquadraCasa[0] + " : "+  
        goalsSquadraCasa[0] + "-" + goalsSquadraOspite[0]);
```

Torino vs. Roma 1-1

Le strutture dati – la classifica

- Utilizzeremo due array *con identico numero di elementi*
- Un array di *String* per i nomi delle squadre
- Un array di *int* per i corrispondenti punteggi delle squadre

```
String[] classNomiSquadre = {"Torino", "Roma", "Napoli"};  
int[] classPuntiSquadre = {0, 0, 0};
```

Le strutture dati – la classifica

- Utilizzeremo due array *con identico numero di elementi*
- Un array di *String* per i nomi delle squadre
- Un array di *int* per i corrispondenti punteggi delle squadre

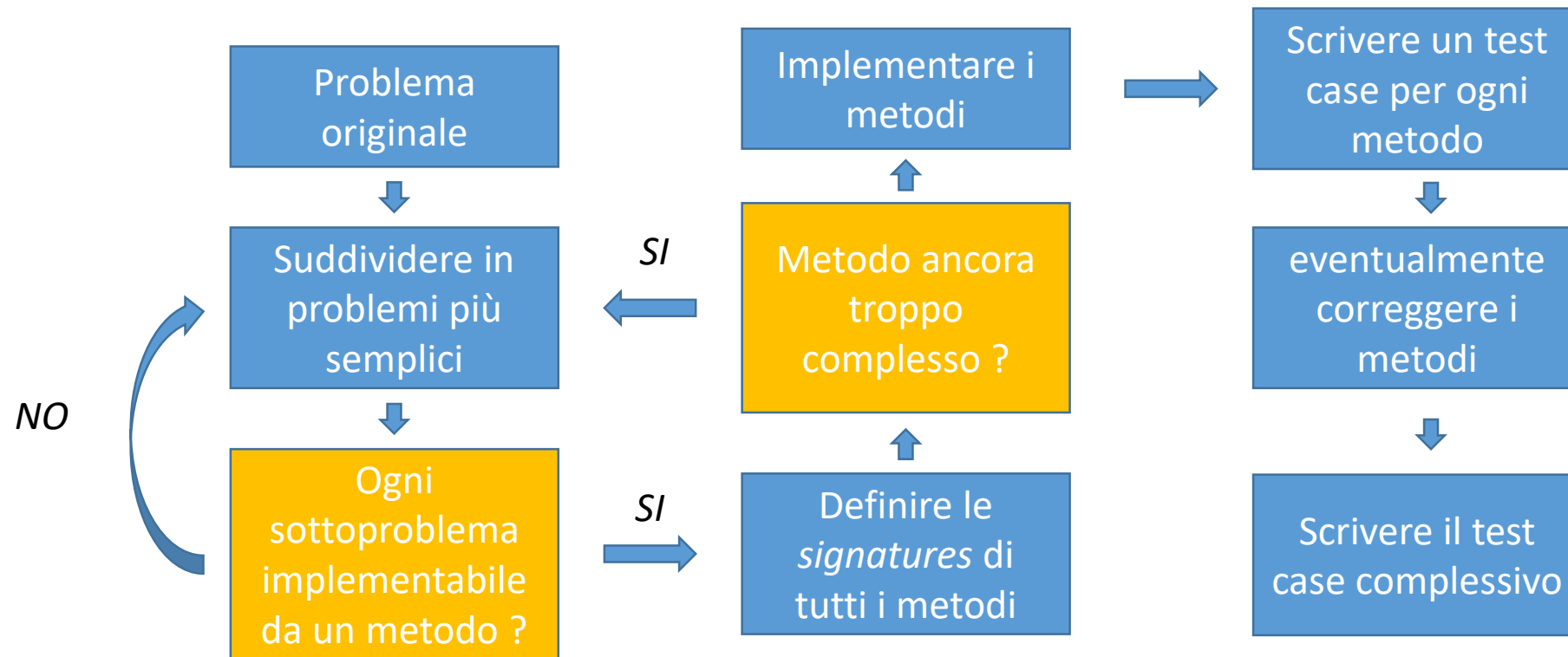
```
String[] classNomiSquadre = {"Torino", "Roma", "Napoli"};  
int[] classPuntiSquadre = {0, 0, 0};
```

- Esempio:

```
println("Squadra: " + classNomiSquadre[i] + " " + classPuntiSquadre[i]);
```

Squadra: Torino 0

I metodi – approccio



I metodi – quanti e quali

- Il file `MetodiTemplate.java` contiene una proposta di metodi
- Problema: analizzare le partite attribuendo i punti in classifica
 - *analizzaPartite(String[], String[], int[], int[], String[], int[])*
 - *analizzaPartita(String[], String[], int[], int[], String[], int[], int)*
 - *incrementaPunti(String[], int[], String, int)*
- Problema: trovare la capolista in classifica non ordinata
 - *int trovaCapolista(int[])*
- Problema: ordinare la classifica (*bubblesort*)
 - *ordinaClassifica(String[], int[])*

Le strutture dati – vincoli

- Quale sarà lo scope degli array ?
 - Di classe -> facilmente accessibili come <nome metodo>.<nome array>
 - Di metodo -> Necessità di portarsi dietro i puntatori ad arrays da metodo a metodo
- Viene imposto di allocare tutti gli array all'interno del main()
- Questo é uno scenario realistico per una vera applicazione

Ordinamento classifica squadre

- Faremo ricorso all'algoritmo *bubblesort*
- *Vedi laboratorio Turno B 15 Ottobre 2020*

ESERCIZIO – Ordinamento con Scambio

```
1  /** OBIETTIVO.  
2   Date quattro variabili a, b, c, d, scrivere un algoritmo che  
3   riorganizzi i valori in esse contenuti, in modo che, al termine,  
4   la variabile d contenga il valore massimo, inizialmente in a, b ,c ,d.  
5  
6   ESEMPIO.  
7   Date le assegnazioni  
8       a = 3; b = 11; c = 8; d = 2;  
9   occorre produrre una configurazione finale tale che:  
10      d==massimo{3, 11, 8, 2}==11  
11   in cui non si mettono vincoli su cosa a, b, c debbano  
12   contenere. */  
13  
14  if (a > b) {  
15      tmp = a; a = b; b = tmp;  
16  }  
17  if (b > c) {  
18      tmp = b; b = c; c = tmp;  
19  }  
20  if (c > d) {  
21      tmp = c; c = d; d = tmp;  
22  }  
23  
24  /* DISPENSE  
25  Sezione 2.2. */
```

ESERCIZIO – Ordinamento con Scambio (1)

```
1 // DOMANDA: modificare il programma sottostante per stampare a schermo anche il numero tot
2 public class OrdinamentoConScambio {
3     public static void main(String[] args) {
4
5         // Valori iniziali delle 4 variabili da ordinare, ignote al momento della scrittura
6         int a = 12; int b = -6; int c = 3; int d = -1;
7         // Variabile di appoggio per lo scambio
8         int tmp;
9         // La variabile binaria di permanenza nel loop si chiamerà eseguiLoop
10        // DOMANDA: perché è inizializzata a true ?
11        boolean eseguiLoop = true;
12        // Contatore di iterazioni strumentale alla stampa di debug senza funzionalità a
13        int contaIterazioni = 1;
14        // La condizione di permanenza nel ciclo while() è eseguiLoop == true
15        while (eseguiLoop == true) {
16            // Stampo la configurazione delle variabili all'inizio del ciclo
17            System.out.println(contaIterazioni + " ) " + a + " " + b + " " + c + " " + d);
18            // Aggiorno il contatore per un'eventuale stampa nel ciclo successivo
19            contaIterazioni ++;
```

ESERCIZIO – Ordinamento con Scambio (2)

```
20
21 //DOMANDA: perché il flag eseguiLoop viene impostato a false incondizionatamen
22 eseguiLoop = false;
23
24 // Scambio a e b nel caso in cui a > b
25 if (a>b) {
26     // Eseguo lo scambio fra a e b
27     tmp = a; a = b; b = tmp;
28     // Imposto il flag a true in caso di scambio effettuato
29     // DOMANDA: perché il flag viene impostato a true solo in caso di scambio
30     eseguiLoop = true;
31 }
32
33 // Valgono considerazioni analoghe al caso sopra
34 if (b>c) {
35     tmp = b; b = c; c = tmp; eseguiLoop = true;
36 }
37 if (c>d) {
38     tmp = c; c = d; d = tmp; eseguiLoop = true;
39 }
40
41
42 }
43
44 }
```

ESERCIZIO – Ordinamento con Scambio

Stato <u>inizio</u> iterazione #	a	b	c	d	flag ulteriore loop
0	12	-6	3	1	t
1	-6	3	1	12	ftt
2	-6	1	3	12	ft
3	-6	1	3	12	f
4	ultimo ciclo senza scambi				

Ordinamento classifica squadre

- Faremo ricorso all'algoritmo *bubblesort*
- *Vedi laboratorio Turno B 15 Ottobre 2020*
- Difficoltà aggiuntive
 - Ordineremo elementi di un array invece che variabili
 - Ordinamento avviene esclusivamente sull' array punti squadre
 - Dovremo ordinare sia l'array punti squadre sia l'array nomi squadre

Ordinamento classifica squadre

- Faremo ricorso all'algoritmo *bubblesort*
- *Vedi laboratorio Turno B 15 Ottobre 2020*
- Difficoltà aggiuntive
 - Ordineremo elementi di un array invece che variabili
 - Scambieremo elementi all'interno di array
 - Ordinamento avviene esclusivamente sull' array punti squadre
 - Dovremo ordinare sia l'array punti squadre sia l'array nomi squadre

Buon lavoro!

