**NOTE PROGRAMMAZIONE 2**

**C++**

STRINGHE

* char \*strcpy(char \*a, const char \*b): copia b su a (compreso il carattere di terminazione di b)
* int strcmp(cost char \*a, const char \*b): restituisce <0 se a<b, 0 se a=b, >0 se a>b
* char \*strcat(char \*a, const char \*b): appende b in coda ad a
* int strlen(const char \*a): restituisce la lunghezza di a

**JAVA**

MEMORY MANAGEMENT

* GC interviene quando serve memoria
* GC elimina gli oggetti per i quali non vi sono più riferimenti attivi
* GC può essere attivato su richiesta esplicita, tramite System.gc()
* operazioni da associarsi con l’eliminnazione di un oggetto possono essere definite nel metodo distruttore finalize()
* il metodo finalize() potrebbe non essere chiamato dal sistema; per essere certi che vengano chiamati i metodi finalize, occorre chiamare la System.runFinalizartion() subito dopo la System.gc()

COSTRUTTORI

* se per una classe A non si scrive alcun costruttore, il sistema crea automaticamente il costruttore A();
* se invece si definisce almeno un costruttore non void, il sistema non crea il costruttore A();
* se B è sottoclasse di A, il costruttore di B come prima cosa invoca A(), a meno che la prima istruzione non sia una super(k); in tal caso non viene chiamato A(), ma A(k)

MODIFICATORI

* abstract  
  - classi abstract non possono essere istanziate e devono essere subclassate  
  - metodi abstract devono essere sovrascritti e possono trovarsi solo in classi abstract
* final  
  - variabili final sono costanti  
  - metodi final non possono essere sovrascritti (non se ne può fare overriding)  
  - classi final non possono essere subclassate
* static: sono variabili e metodi associati a una classe, piuttosto che a un oggetto  
  - una variabile static è una variabile condivisa tra tutte le istanze della classe  
  - un metodo static lavora solo con variabili static e può essere chiamato senza creare un’istanza
* public: visibile da tutti
* nessuno: visibile nel package
* protected: visibile nel package e dalle sottoclassi
* private: nascosta da tutti (visibile solo all’interno della classe)

STATIC BINDING vs DYNAMIC BINDING

Cosa succede quando si chiama un metodo su un oggetto? (Esempio: C obj; … obj.f(args))

1. Il compilatore cerca tra i metodi dell’oggetto obj i metodi f(…) e li enumera
2. Il compilatore determina i tipi di parametri passati; se trova un match unico con i metodi della classe C (anche tramite cast) ne prende nota (overloading resolution), altrimenti genera un messaggio di errore
3. Se il metodo è private, static, final o un costruttore il compilatore sa esattamente che metodo chiamare (static binding)
4. Altrimenti il metodo dipende da qual è la classe a cui obj appartiene a runtime, e il compilatore deve delegare alla JVM la determinazione del metodo a run time (dynamic binding)

**JAVA FX**

Tipi di Parent (Parent è una classe abstract):

* Control
  + - Label
    - TextField
* Group
* Region
  + - Pane
      * + AnchorPane
        + BorderPane
        + TilePane
        + GridPane
        + StackPane
        + HBox
        + VBox
        + FlowPane
* WebRegion

Tipi di Shape:

* Line
* Polyline
* Polygon
* Rectangle
* Arc
* Circle
* Ellipse
* QuadCurve
* CubicCurve
* Text

Ascoltatori

* Listener esterno: usato quando deve ascoltare eventi che provengono da classi diverse
* Listener interno: usato per ascoltare gli eventi della classe in cui si trova  
  può essere interno o interno anonimo
* Self listener