SAPIENZA
UNIVERSITÀ DI ROMA
DIPARTIMENTO DI INFORMATICA

Metodologie di Programmazione

- Lezione 2: Hello World!

Esercizi

- Scrivere una classe Java chiamata PrimoProgramma che stampi a video la stringa "Primo Programma", andando poi a capo
- Scrivere una classe Java chiamata SecondoProgramma che stampi a video la stringa "Hai digitato: " seguita dal primo parametro in input (args[0]). Si utilizzino i metodi print e println
- Svolgere nuovamente il secondo esercizio utilizzando il metodo printf

UNITELMA SAPIENZA

SAPIENZA
UNIVERSITÀ DI ROMA
DIPARTIMENTO DI INFORMATICA

Metodologie di Programmazione

- Lezione 3: Tipi di dato fondamentali

Esercizi

1. Scrivere l'istruzione Java che definisce una variabile intera k con valore 42
2. Scrivere l'istruzione necessaria a definire una variabile s di tipo stringa assegnandole il valore "ciao"
3. Modificare l'esercizio 2 assegnando alla stringa s la prima stringa fornita in input dall'utente al metodo main

SAPIENZA
UNIVERSITÀ DI ROMA
DIPARTIMENTO DI INFORMATICA

Metodologie di Programmazione

- Lezione 4: Conversioni di tipo

Esercizi (1)

1. Scrivere una classe Moltiplica che, dati in input 2 numeri interi, ne restituisca a video il prodotto
2. Scrivere una classe StampaNome che, dato in input un nome, lo stampi tra due righe di trattini. Ad es.:

```
+---------+
Roberto
+---------+
```

3. Scrivere una classe Somma10 che calcoli la somma dei primi dieci interi positivi

Esercizi (2)

4. Scrivere una classe Variabili che, all'interno del metodo main, dichiari una variabile intera i, una variabile di tipo stringa s e una variabile double d. Quindi vengono svolte le seguenti tre operazioni:
   - La stringa viene inizializzata al valore del primo argomento fornito in input al main
   - All'intero viene assegnato il valore intero della stringa
   - Al double viene assegnata la metà del valore di i (ad es. se i è pari a 3, d sarà pari a 1.5)
   - I valori di s, i e d vengono stampati a video

SAPIENZA
UNIVERSITÀ DI ROMA
DIPARTIMENTO DI INFORMATICA

Metodologie di Programmazione

Lezione 6: Concetti fondamentali della programmazione orientata agli oggetti

Esercizio:
la classe Rettangolo

UNITELMA SAPIENZA
SAPIENZA UNIVERSITÀ DI ROMA
DIPARTIMENTO DI INFORMATICA

- Progettare una classe Rettangolo i cui oggetti rappresentano un rettangolo e sono costruiti a partire dalle coordinate x, y e dalla lunghezza e altezza del rettangolo
- La classe implementa i seguenti metodi:
  - trasla che, dati in input due valori x e y, trasla le coordinate del rettangolo dei valori orizzontali e verticali corrispondenti
  - toString che restituisce una stringa del tipo "(x1, y1)-> (x2, y2)" con i punti degli angoli in alto a sinistra e in basso a destra del rettangolo
- Implementare una classe di test TestRettangolo che verifichi il funzionamento della classe Rettangolo sul rettangolo in posizione (0, 0) e di lunghezza 20 e altezza 10, traslandolo di (10, 5) (ovvero 10 verso destra e 5 in basso)

Esercizio:
la classe ContoCorrente

UNITELMA SAPIENZA
SAPIENZA UNIVERSITÀ DI ROMA
DIPARTIMENTO DI INFORMATICA

- Si progetti la classe ContoCorrente che rappresenta il conto corrente di un cliente. Un oggetto della classe è costruito a partire dall'identificativo del cliente e, opzionalmente, dal valore iniziale del conto corrente (altrimenti pari a 0). La classe implementa i seguenti metodi:
  - svuota: il valore del conto viene azzerato e il quantitativo di denaro presente viene restituito in output all'utente;
  - getImporto: restituisce l'attuale importo del conto;
  - getIdUtente: restituisce l'identificativo dell'utente;
  - preleva: preso in input un importo da prelevare, modifica l'importo del conto in modo da considerare tale prelievo di denaro;
  - versa: incrementa l'importo del conto della quantità di denaro passata in ingresso.

SAPIENZA
UNIVERSITÀ DI ROMA
DIPARTIMENTO DI INFORMATICA

Metodologie di Programmazione

Lezione 7: Incapsulamento e inizializzazione di default

Esercizio: registratore di cassa

UNITELMA SAPIENZA
SAPIENZA UNIVERSITÀ DI ROMA
DIPARTIMENTO DI INFORMATICA

- Progettare una classe che costituisca un modello di registratore di cassa
- La classe deve consentire a un cassiere di inserire i prezzi di articoli e la quantità di denaro pagata dal cliente, calcolando il resto dovuto
- La classe fornisce i seguenti metodi:
  - Registra il prezzo di vendita per un articolo
  - Registra la somma di denaro pagata finora
  - Conclude la transazione, calcola il resto dovuto al cliente nel momento e restituendolo in uscita

SAPIENZA
UNIVERSITÀ DI ROMA
DIPARTIMENTO DI INFORMATICA

Metodologie di Programmazione

Lezione 8: La classe String

Esercizio:
la Stringa "magica"

UNITELMA SAPIENZA
SAPIENZA
UNIVERSITÀ DI ROMA
DIPARTIMENTO DI INFORMATICA

- Progettare una classe Stringa42, costruita a partire da 3 stringhe in input, che concateni le stringhe inframezzate da spazi e conservi solo i primi 42 caratteri della stringa risultante
- La classe deve poter:
  - restituire la stringa di lunghezza massima 42
  - restituire l'iniziale di tale stringa
  - restituire un booleano che indichi se la stringa è pari al numero "magico" 42
  - restituire un booleano che indichi se la stringa contiene il numero "magico" 42

Esercizio: punti e segmenti

UNITELMA SAPIENZA
SAPIENZA UNIVERSITÀ DI ROMA
DIPARTIMENTO DI INFORMATICA

- Progettare una classe Punto per la rappresentazione di un punto nello spazio tridimensionale
- E una classe Segmento per rappresentare un segmento nello spazio tridimensionale
- Scrivere una classe di test che crei:
  - due oggetti della classe Punto con coordinate (1, 3, 8) e (4, 4, 7)
  - un oggetto della classe Segmento che rappresenti il segmento che unisce i due punti di cui sopra

SAPIENZA
UNIVERSITÀ DI ROMA
DIPARTIMENTO DI INFORMATICA

Metodologie di Programmazione

Lezione 9: Riferimenti a oggetti, heap & stack

Esercizio: punti e segmenti

UNITELMA SAPIENZA
SAPIENZA UNIVERSITÀ DI ROMA
DIPARTIMENTO DI INFORMATICA

- Progettare una classe Punto per la rappresentazione di un punto nello spazio tridimensionale
- E una classe Segmento per rappresentare un segmento nello spazio tridimensionale
- Scrivere una classe di test che crei:
  - due oggetti della classe Punto con coordinate (1, 3, 8) e (4, 4, 7)
  - un oggetto della classe Segmento che rappresenti il segmento che unisce i due punti di cui sopra
- Raffigurare l'evoluzione dello stato della memoria, distinguendo tra stack e heap

SAPIENZA
UNIVERSITÀ DI ROMA
DIPARTIMENTO DI INFORMATICA

Metodologie di Programmazione

Lezione 10: Strutture di controllo e costrutti iterativi

Esercizio: conta parola

- Scrivere un metodo che, presi in ingresso un testo sotto forma di stringa e una parola w, trasformi il testo in parole (token) e conti le occorrenze di w nel testo

```
public int contaParola(String testo, String w)
{
    int cont = 0;
    String[] tokens = testo.split(" ");

    for (int k = 0; k < tokens.length; k++)
        if (tokens[k].equals(w)) cont++;

    return cont;
}
```

Esercizio: stringa verticale

- Scrivere un metodo che legge una stringa da console e la stampa in verticale un carattere per linea
- Ad esempio, dato in input "ciao", viene stampato:

```
c
i
a
o
```

Esercizio: stringhe verticali

- Scrivere un metodo che riceve tre stringhe e le stampa in verticale una accanto all'altra
- Ad esempio: date "ciao", "buondì", "hello", stampa:

```
cbh
iue
aol
onl
 do
 ì
```

Esercizio: conta vocali

- Scrivere un metodo che riceve una stringa e stampa a video il conteggio delle vocali in essa contenute
- Ad esempio: data la stringa "le aiuole sono pulite", il metodo stampa:

```
a=1 e=3 i=2 o=3 u=2
```

Esercizio: divisori

- Scrivere un metodo che, dato un intero positivo n in ingresso, stampi i divisori propri di n (ovvero i divisori < n)
- Ad esempio, dato l'intero 20, il metodo stampa:

```
1, 2, 4, 5, 10
```

Esercizio: somma dei due numeri precedenti

- Scrivere un metodo che, dati in ingresso due interi a e b e un terzo intero N, stampi a e b e gli N numeri della sequenza in cui ogni numero è la somma dei due precedenti
- Ad esempio, dati gli interi 2, 3 e 6, il metodo stampa:

```
2, 3, 5, 8, 13, 21, 34, 55
```

Esercizio: conversione dei numeri

UNITELMA SAPIENZA
SAPIENZA UNIVERSITÀ DI ROMA
DIPARTIMENTO DI INFORMATICA

- Progettare una classe per la conversione di base dei numeri interi
- Ogni oggetto della classe viene costruito con un intero o con una stringa che contiene l'intero
- La classe è in grado di convertire l'intero nella base desiderata (restituito sotto forma di stringa)

Esercizio: frase palindroma

UNITELMA SAPIENZA
SAPIENZA UNIVERSITÀ DI ROMA
DIPARTIMENTO DI INFORMATICA

- Una stringa è palindroma se rimane uguale quando viene letta da sinistra verso destra o da destra verso sinistra
- Scrivere un metodo che dica che una stringa è palindroma
- Scrivere anche una classe di collaudo che testi il metodo in diverse situazioni
- Ad esempio, data in ingresso le stringhe "angelalavalalegna" o "itopinonavevanonipoti", il metodo deve segnalare che la stringa è palindroma

Esercizio:
la classe Cornice

- Progettare una classe Cornice che rappresenti una cornice NxN, con eventuale stringa contenuta al suo interno
- La cornice deve disporre del metodo toString() che ne restituisce la rappresentazione in formato stringa
- Ad esempio: new Cornice(6, "Cornici in Java").toString() restituisce la seguente stringa

```
******
*Corn*
*ici *
*in J*
*ava *
******
```

Esercizio: terne pitagoriche

- Una terna pitagorica è una tripla di numeri interi a, b, c tali che 1 <= a <= b <= c e $a^2+b^2 = c^2$
  - Ovvero a e b sono i lati di un triangolo rettangolo e c l'ipotenusa
- Scrivere un metodo che legge un intero N e stampa tutte le triple pitagoriche con c <= N
- Ad esempio: dato N=15 il metodo stampa:

```
a=3 b=4 c=5
a=6 b=8 c=10
a=5 b=12 c=13
a=9 b=12 c=15
```

Esercizio: da cifre a lettere e viceversa

- Scrivere un metodo che prenda in ingresso una stringa contenente cifre e restituisca una stringa in cui ciascuna cifra è stata trasformata nella parola corrispondente
- Ad esempio, data in input la stringa "8452", il metodo restituisce "otto quattro cinque due"
- Viceversa, scrivere un metodo che prenda in ingresso una stringa contenente cifre scritte a lettere e restituisca una stringa contenente le cifre corrispondenti
- Ad esempio, data in input la stringa "otto quattro cinque due", il metodo restituisce "8452"

Esercizio: stampa triangoli

- Scrivere un metodo che, dato un intero positivo dispari N, stampi un triangolo isoscele la cui base è costituita da N caratteri
- Ad esempio, dato l'intero 5, il metodo stampa:

```
  *
 ***
*****
```

Esercizio: la classe RettangoloDiCaratteri

UNITELMA SAPIENZA

SAPIENZA
UNIVERSITÀ DI ROMA
DIPARTIMENTO DI INFORMATICA

- Progettare una classe RettangoloDiCaratteri che rappresenta un rettangolo riempito con caratteri *
- Un oggetto della classe viene costruito fornendo la posizione x, y e la lunghezza e altezza del rettangolo
- Il metodo draw si occupa di stampare il rettangolo a video, partendo dalla posizione (0,0)
- Ad esempio, dato il rettangolo (2, 2, 4, 3), il metodo draw() stampa (␣ rappresenta una spazio):

```
␣
␣
␣␣****
␣␣****
␣␣****
```

Esercizio: ampliare la classe RettangoloDiCaratteri

UNITELMA SAPIENZA

SAPIENZA
UNIVERSITÀ DI ROMA
DIPARTIMENTO DI INFORMATICA

- Aggiungere alla classe RettangoloDiCaratteri i seguenti metodi:
  - setCarattere(): Permette di specificare il carattere da utilizzare per stampare i rettangoli
  - drawVerticalStripes(): stampa il rettangolo a strisce verticali usando anche un secondo carattere, ad esempio:

```
*$*$
*$*$
*$*$
```

  - drawHorizontalStripes(): stampa il rettangolo a strisce orizzontali
  - drawChessboard(): stampa il rettangolo a mo' di scacchiera:

```
*$*$
$*$*
*$*$
```

  - Una seconda versione di setCarattere() che permetta di specificare entrambi i caratteri da utilizzare per la stampa
  - Un metodo che permetta di modificare la posizione del rettangolo
  - Un metodo che permetta di accedere ai due caratteri usati per la stampa

SAPIENZA
UNIVERSITÀ DI ROMA
DIPARTIMENTO DI INFORMATICA

Metodologie di Programmazione

Lezione 11: Gli array

Esercizi semplici con gli array

UNITELMA SAPIENZA
SAPIENZA UNIVERSITÀ DI ROMA
DIPARTIMENTO DI INFORMATICA

- Scrivere un metodo che, dato un array di stringhe e una stringa in input, restituisca true se l'array contiene la stringa, false altrimenti
- Scrivere una seconda versione del metodo che restituisca la posizione della stringa trovata, -1 altrimenti
- Scrivere un metodo che, dato un array di double, restituisca il valore massimo dell'array

Esercizio: reimplementa conta vocali

UNITELMA SAPIENZA
SAPIENZA UNIVERSITÀ DI ROMA
DIPARTIMENTO DI INFORMATICA

- Scrivere un metodo che riceve una stringa e stampa a video il conteggio delle vocali in essa contenute
  - Utilizzare un array per il conteggio separato delle 5 vocali
- Ad esempio: data la stringa "le aiuole sono pulite", il metodo stampa:

```
a=1 e=3 i=2 o=3 u=2
```

- Come l'avevate implementato **SENZA** array?

Esercizio: implementare un filtro

- Progettare una classe Filtro costruita con un array di interi
- La classe implementa operazioni che permettono di ottenere nuovi sotto-array dell'array iniziale:
  - passaBasso(): restituisce tutti gli elementi <= k nell'ordine iniziale
  - passaAlto(): restituisce tutti gli elementi >= k nell'ordine iniziale
  - filtra(): restituisce l'array iniziale da cui sono state eliminate tutte le occorrenze dell'intero passato in input
  - filtra(): una seconda versione del metodo che restituisce l'array iniziale da cui vengono eliminate tutte le occorrenze di interi presenti nell'array passato in input
- Se Filtro viene costruito con l'array { 1, 2, 10, 2, 42, 7, 8 }:
  - passaBasso(8) restituisce { 1, 2, 2, 7, 8 }
  - passaAlto(9) restituisce { 10, 42 }
  - filtra(2) restituisce { 1, 10, 42, 7, 8 }
  - filtra(new int[] { 2, 7, 42 }) restituisce { 1, 10, 8 }

Esercizio: reimplementare Arrays.copyOf() e Arrays.toString()

- Implementare un metodo statico copyOf che, analogamente a java.util.Arrays.copyOf, copi un array in un nuovo array delle dimensioni specificate (troncando l'array in input, se più grande)

Esercizio: sequenza di cifre estensibile

UNITELMA SAPIENZA
SAPIENZA UNIVERSITÀ DI ROMA
DIPARTIMENTO DI INFORMATICA

- Progettare una classe SequenzaDiCifre che espone un metodo che, data in input una stringa e un intero N, aggiunga alla sequenza inizialmente vuota (rappresentata mediate un array) le prime N cifre contenute nella stringa (si assuma che ne contenga comunque almeno N). La classe espone anche un metodo toString che fornisce una rappresentazione sotto forma di stringa della sequenza.
- Ad esempio:

```
SequenzaDiCifre s = new SequenzaDiCifre();
s.aggiungiCifre("abc1--23", 2);
s.aggiungiCifre("xx0a8b76543100", 4);
System.out.println(s.toString());
```

- stampa: [1,2,0,8,7,6]

Esercizio:
array con fattoriale

- Scrivere un metodo che, dato un intero n in ingresso, restituisca un array di dimensione n contenente k! nella k-esima cella, per ogni valore di k

```
public int[] getArrayFattoriali(final int n)
{
    int[] fatt = new int[n];

    fatt[0] = 1;
    for (int k = 1; k < n; k++) fatt[k] = fatt[k-1]*k;

    return fatt;
}
```

Esercizio: implementare una lista mediante array

UNITELMA SAPIENZA
SAPIENZA UNIVERSITÀ DI ROMA
DIPARTIMENTO DI INFORMATICA

- Che cos'è una lista? E' una sequenza di oggetti
- Implementare una classe ListaDiInteri che permetta le seguenti operazioni:
  - Restituisce l'elemento i-esimo della lista
  - Restituisce l'indice della posizione dell'intero fornito in input
  - Restituisce una stringa formattata contenente la lista di interi
  - Restituisce la dimensione della lista
  - Contiene un determinato intero (true o false)?
  - Aggiunge un intero in coda alla lista
  - Aggiunge un intero nella posizione specificata
  - Elimina la prima occorrenza di un intero dalla lista
  - Elimina l'elemento i-esimo della lista

Esercizio: stampa di istogrammi

UNITELMA SAPIENZA
SAPIENZA UNIVERSITÀ DI ROMA
DIPARTIMENTO DI INFORMATICA

- Progettare una classe Istogramma che rappresenta la distribuzione di dati (es. voti degli studenti) in un intervallo da i a j fornito in input (es. da 0 a 31 (trenta e lode))
- La classe permette di incrementare il conteggio in corrispondenza di ciascun elemento dell'intervallo (es. memorizzando così un nuovo voto di uno studente)
- La classe può stampare a video l'istogramma corrispondente
  - Più facile in orizzontale
  - Provate a stampare in verticale!!!

Esercizio: mescolare e distribuire un mazzo di carte da gioco

UNITELMA SAPIENZA

SAPIENZA
UNIVERSITÀ DI ROMA
DIPARTIMENTO DI INFORMATICA

- Progettare una classe Carta che rappresenti una singola carta da gioco (con seme e valore)
- La classe deve restituire su richiesta la propria rappresentazione sotto forma di stringa
- Progettare quindi una classe MazzoDiCarte che rappresenti un intero mazzo da 52 carte
- La classe deve implementare i seguenti metodi:
  - mescola il mazzo di carte
  - distribuisci la prossima carta
- Infine si progetti una classe di collaudo che crea un mazzo, mescoli le carte e ne distribuisca carte fino ad esaurimento del mazzo

Esercizio:
la tavola pitagorica

- Scrivere una classe che rappresenti la tavola pitagorica NxN (dove l'intero N è un parametro di costruzione della classe)
- La classe deve, su richiesta, restituire il valore della tabella in corrispondenza della posizione (i, j)
- La classe deve poter stampare l'intera tavola

x	0	1	2	3	4	5	6	7	8	9	10
0	0	0	0	0	0	0	0	0	0	0	0
1	0	1	2	3	4	5	6	7	8	9	10
2	0	2	4	6	8	10	12	14	16	18	20
3	0	3	6	9	12	15	18	21	24	27	30
4	0	4	8	12	16	20	24	28	32	36	40
5	0	5	10	15	20	25	30	35	40	45	50
6	0	6	12	18	24	30	36	42	48	54	60
7	0	7	14	21	28	35	42	49	56	63	70
8	0	8	16	24	32	40	48	56	64	72	80
9	0	9	18	27	36	45	54	63	72	81	90
10	0	10	20	30	40	50	60	70	80	90	100

Esercizio: il gioco del tris

UNITELMA SAPIENZA

SAPIENZA
UNIVERSITÀ DI ROMA
DIPARTIMENTO DI INFORMATICA

- Progettare una classe ScacchieraTris che implementi la scacchiera del gioco del tris
- La classe deve memorizzare la scacchiera i cui elementi possono essere:
  - " " (se non è stata ancora occupata la casella)
  - "X" oppure "O" (secondo il giocatore che ha occupato la casella)
- La classe deve stampare in qualsiasi momento la situazione della scacchiera
- Deve permettere di occupare una casella con un simbolo "X" o "O"
- Progettare quindi una classe Tris che implementi il gioco utilizzando la scacchiera appena progettata

SAPIENZA
UNIVERSITÀ DI ROMA
DIPARTIMENTO DI INFORMATICA

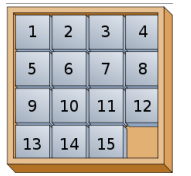
Metodologie di Programmazione

Lezione 12: Static, consistenza ed enumerazioni

Esercizio: Campo Minato

- Progettare la classe CampoMinato che realizzi il gioco del campo minato (http://it.wikipedia.org/wiki/Campo_minato_(videogioco))
- Il costruttore deve inizializzare il campo NxM (dove N e M sono interi forniti in ingresso al costruttore insieme al numero m di mine) piazzando casualmente le m mine nel campo
- Implementare un metodo scopri() che, dati x e y in ingresso, scopre la casella e restituisce un intero pari a:
  - -1 se la casella contiene una mina
  - La quantità di caselle adiacenti contenenti mine (incluse quelle in diagonale)
  - 0 se la caselle adiacenti non contengono mine. In quest'ultimo caso, vengono scoperte anche le caselle adiacenti **finché** non si incontra un numero > 0 (richiede la **ricorsione**!)
- Implementare un metodo toString() che restituisce la situazione attuale del gioco
- Implementare un metodo vinto() che restituisce lo **stato del gioco**: perso, vinto, in gioco

Esercizio:
Gioco del Quindici



- Progettare la classe GiocoDelQuindici che realizzi il gioco del quindici (http://it.wikipedia.org/wiki/Gioco_del_quindici)
- Il costruttore deve inizializzare una tabellina 4x4 in cui sono posizionate casualmente 15 tessere quadrate (da 1 a 15)
- Implementare un metodo privato mischia che posiziona le caselle casualmente nella tabella (usato anche dal costruttore)
- Implementare un metodo scorri che prende in ingresso la posizione x e y della casella e la **direzione** in cui spostare la casella
- Implementare un metodo vinto che restituisce un booleano corrispondente alla vincita del giocatore (ovvero se si è riuscito a posizionare le caselle esattamente nell'ordine da 1 a 15, come riportato in alto a destra in figura)

SAPIENZA
UNIVERSITÀ DI ROMA
DIPARTIMENTO DI INFORMATICA

Metodologie di Programmazione

Lezione 14: Ereditarietà (parte 2)

BarraDiEnergia & BarraConPercentuale

- Creare una classe BarraDiEnergia costruita con un intero k che ne determina la lunghezza massima. Inizialmente la barra è vuota. La barra è dotata di un metodo per l'incremento unitario del suo livello di riempimento e di un metodo toString che ne fornisca la rappresentazione sotto forma di stringa (es. se il livello è 3 su 10, la stringa sarà "OOO=======".
- Creare quindi una seconda classe BarraDiEnergiaConPercentuale che fornisce una rappresentazione sotto forma di stringa come BarraDiEnergia ma stampando in coda alla stringa la percentuale del livello di riempimento. Per esempio, se il livello è 3 su 10, la stringa sarà "OOO======= 30%".

Esercizio: ListaDiInteri e ListaOrdinataDiInteri

UNITELMA SAPIENZA
SAPIENZA UNIVERSITÀ DI ROMA
DIPARTIMENTO DI INFORMATICA

- Implementare una classe ListaDiInteri mediante un array (con i metodi specificati in fondo alle diapositive della parte: "controllo e array")
- Implementare quindi una classe ListaOrdinataDiInteri per creare liste di interi ordinati in modo crescente. Oltre agli altri metodi di ListaDiInteri (esclusi quelli di aggiunta) essa definisce i seguenti 3 metodi di aggiunta:
  - Aggiungi un intero in coda alla lista: l'aggiunta avviene solo se l'intero preserva l'ordine della lista.
  - Aggiungi un intero nella posizione specificata: come sopra, l'aggiunta avviene solo se l'intero preserva l'ordine degli interi della lista
  - Aggiungi un intero: l'intero viene inserito nella posizione appropriata, in modo da preservare l'ordine degli interi della lista
- L'array non deve essere ordinato con metodi di sorting, quali Arrays.sort (né vostri metodi di sorting *completo* dell'array)
- **Extra:** permettere di specificare un parametro da passare opzionalmente al costruttore di ListaOrdinataDiInteri per stabilire l'ordine della lista (crescente o decrescente; per default, crescente)

Esercizio: Animali

- Progettare la classe Animale che rappresenti un generico animale
- La classe possiede i metodi emettiVerso() e getNumeroDiZampe()
- Possiede inoltre il metodo getTaglia() che restituisce un valore scelto tra: piccola, media e grande.
- Progettare quindi le classi Mammifero, Felino, Gatto (taglia piccola), Tigre (grande), Cane, Chihuahua (piccola), Beagle (media), Terranova (grande), Uccello, Corvo (media), Passero (piccola), Millepiedi (piccola)
- Personalizzare in modo appropriato la taglia, il numero di zampe e il verso degli animali

Esercizio: Conto bancario

UNITELMA SAPIENZA

SAPIENZA
UNIVERSITÀ DI ROMA
DIPARTIMENTO DI INFORMATICA

- Progettare la classe ContoBancario che rappresenti un conto con informazioni relative al denaro attualmente disponibile, il codice IBAN
- Modellare quindi una generica operazione bancaria Operazione che disponga di un metodo esegui()
- Modellare quindi i seguenti tipi di operazione:
  - PrelevaDenaro: preleva una specificata quantità di denaro da un dato conto
  - SvuotaConto: preleva tutto il denaro da un dato conto
  - VersaDenaro: versa del denaro sul conto specificato
  - SituazioneConto: stampa l'attuale saldo del conto
  - Bonifico: preleva del denaro da un conto e lo versa su un altro

Esercizio: Distributore di bevande

UNITELMA SAPIENZA
SAPIENZA UNIVERSITÀ DI ROMA
DIPARTIMENTO DI INFORMATICA

- Progettare una classe Prodotto con un prezzo e tre tipi diversi di prodotto: Caffè, Cappuccino, Cioccolato
- Progettare la classe DistributoreDiBevande che rappresenti un distributore automatico costruito con un intero N che determina il numero di prodotti nel distributore
- La classe prevede i seguenti metodi:
  - un metodo carica() che inserisce N prodotti di tipo e ordine casuale
  - un metodo inserisciImporto() che permette di inserire un importo nella macchinetta
  - un metodo getProdotto() che, dato in ingresso un numero di prodotto, restituisca il prodotto associato a quel numero e decrementi il saldo disponibile nel distributore
  - un metodo getSaldo() che restituisca il saldo attuale del distributore
  - un metodo getResto() che restituisca il resto dovuto e azzeri il saldo

Esercizio: Espressioni matematiche

UNITELMA SAPIENZA — SAPIENZA Università di Roma, Dipartimento di Informatica

- Progettare una serie di classi che modellino le espressioni matematiche secondo la seguente definizione:
  - Una costante di tipo double è un'espressione
  - Una variabile con nome di tipo stringa e valore double è un'espressione
  - Se e_1 è un'espressione, allora $-e_1$ è un'espressione
  - Se e_1, e_2 sono espressioni, allora e_1 op e_2 è un'espressione dove op può essere l'operatore +, -, *, /, %
- Ogni tipo di espressione (costante, variabile, espressioni composte) deve essere modellata mediante una classe separata
- Ogni espressione dispone del metodo getValore() che restituisce il valore che quell'espressione possiede in quel momento
- Costruire quindi l'espressione –(5+(3/2)-2)*x e calcolarne il valore quando la variabile x vale 3 e quando la variabile x vale 6

Esercizio: il Gioco dell'Oca

UNITELMA SAPIENZA
SAPIENZA UNIVERSITÀ DI ROMA
DIPARTIMENTO DI INFORMATICA

- Progettare il Gioco dell'Oca modellando:
  - Il Giocatore, che mantiene l'informazione sulla posizione nel tabellone e i punti accumulati e implementa il metodo tiraDadi()
  - Il Tabellone come sequenza di caselle costruita a partire da un intero N e da un elenco di giocatori; la classe dispone dell'operazione di posizionamento dei giocatori tenendo conto dell'effetto "gioco dell'oca" in cui, arrivati alla fine, si torna indietro
- Diversi tipi di caselle ciascuna con un diverso effetto a seguito del posizionamento del giocatore su quella casella:
  - una CasellaVuota (nessun effetto sul giocatore)
  - una CasellaSpostaGiocatore che sposta il giocatore di x caselle (avanti se x > 0 o indietro se x < 0)
  - una CasellaPunti che ha l'effetto di far guadagnare o perdere un certo numero di punti al giocatore
  - la classe GiocoDellOca che, dato un intero N e dati i giocatori, che inizializza un tabellone di lunghezza N e implementa il metodo giocaUnTurno() che fa effettuare una mossa a ognuno dei giocatori

Esercizio: Tetris

- Progettare il gioco del Tetris modellando la classe Pezzo con le seguenti operazioni:
  - Left : sposta a sinistra il pezzo
  - Right: sposta a destra il pezzo
  - Rotate: ruota il pezzo in senso orario
  - Down: manda giù il pezzo
- Progettare anche la classe di ciascun pezzo (a forma di L, a forma di T, a serpente, a forma di I e cubo)
- Progettare infine la classe Tetris che somministra i pezzi, permette al giocatore di muoverli, gestisce lo spazio libero e calcola i punteggi del giocatore

SAPIENZA
UNIVERSITÀ DI ROMA
DIPARTIMENTO DI INFORMATICA

Metodologie di Programmazione

Lezione 17: Esercizi su ereditarietà e polimorfismo

Esercizio:
la LibreriaComponibile

UNITELMA SAPIENZA
SAPIENZA UNIVERSITÀ DI ROMA
DIPARTIMENTO DI INFORMATICA

- Progettare una libreria componibile, costituita da una serie di scaffali
- Ogni scaffale contiene una sequenza di libri e prevede una capienza massima
- Un libro è rappresentato dal titolo e dall'autore
- Una libreria permette di aggiungere ed eliminare scaffali, di accedere al k-esimo scaffale e di ottenere il numero di scaffali
- Uno scaffale permette di aggiungere libri, eliminare libri per titolo e cercare libri per titolo
- La libreria inoltre permette di aggiungere un libro nel primo scaffale libero

Esercizio: RiproduttoreMusicale (1)

- Progettare una classe RiproduttoreMusicale che rappresenti un generico riproduttore
- La classe deve realizzare i seguenti metodi:
  - inserisciSupporto(): permette di inserire un supporto musicale (es. CD, nastro, ecc.)
  - espelliSupporto(): espelle il supporto
  - getBrano(): restituisce l'attuale brano in esecuzione (null se non sta eseguendo)
  - play(): esegue il brano attualmente selezionato
  - stop(): interrompe l'esecuzione
  - next(): seleziona il prossimo brano
  - prev(): seleziona il brano precedente
  - toString(): visualizza le informazioni del brano attualmente in esecuzione

Esercizio: RiproduttoreMusicale (2)

UNITELMA SAPIENZA
SAPIENZA UNIVERSITÀ DI ROMA
DIPARTIMENTO DI INFORMATICA

- Realizzare quindi i seguenti riproduttori
  - Giradischi
  - Mangianastri
  - Lettore CD
  - Lettore Mp3
- Realizzare inoltre diversi tipi di supporto:
  - Disco a 33 giri (14 brani)
  - Disco a 45 giri (2 brani)
  - Compact Disc (20 brani)
  - Nastro (con un numero specificato di minuti, numero di brani pari al numero di minuti / 5)
  - Memoria USB (1024 brani)
- Gli ultimi due supporti permettono di registrare/inserire brani nella posizione specificata

Esercizio: RiproduttoreMusicale (3)

UNITELMA SAPIENZA

SAPIENZA
UNIVERSITÀ DI ROMA
DIPARTIMENTO DI INFORMATICA

- Ciascun supporto può contenere il numero massimo di brani specificati tra parentesi nella precedente diapositiva e viene costruito con una data sequenza (eventualmente vuota) di brani

- Modellare inoltre la classe Brano che contenga l'informazione sul nome del brano e il cantante

Esercizio: Il labirinto

UNITELMA SAPIENZA

- Progettare una classe Labirinto che r
un labirinto
- La classe ha un unico punto di acces
singolo corridoio che viene reso disponibile al
giocatore
- Ogni corridoio ha un numero variabile di punti di accesso ad altrettanti corridoi
- Un corridoio implementa un metodo che determina se il giocatore ha raggiunto il corridoio d'uscita del labirinto
- A ogni passo, il giocatore deve scegliere uno dei possibili corridoi accessibili dall'attuale posizione

Esercizio: Esseri viventi (1)

- Progettare una classe EssereVivente dotata di età, sesso (M, F) e nome, con i seguenti metodi:
  - getEta(): restituisce l'età dell'essere
  - getSesso(): restituisce il sesso dell'essere
  - getNome(): restituisce il nome
  - cresce(): fa crescere l'essere vivente di 1 anno e con una certa probabilità lo fa morire
  - mangia(): fa mangiare l'essere vivente
  - siRiproduceCon(EssereVivente e): dato in ingresso un altro EssereVivente, se di sesso opposto fa riprodurre i due esseri e ne crea altri, altrimenti emette eccezione
  - muore(): causa la morte dell'essere vivente; a seguito di questa operazione, l'essere non può eseguire nessun'altra operazione

Esercizio: Esseri viventi (2)

- Implementare i seguenti tipi di esseri viventi:
  - EssereUmano: rappresenta un essere umano: si riproduce in un numero di 1 o 2 esemplari
  - Coniglio: rappresenta un coniglio, che si riproduce in al più 10 esemplari
  - Fenice: rappresenta l'uccello mitologico che resuscita dalle ceneri dopo la morte; si preveda quindi un metodo risorgi() che ha l'effetto di far rinascere la fenice
  - Gatto: rappresenta un gatto, che si riproduce in al più 5 esemplari, e che possiede 7 vite (ogni volta che muore, può accedere alla vita successiva)
  - Pesce: rappresenta un pesce, che si riproduce tra 30 e 100 esemplari
  - PescePagliaccio: rappresenta una specie di pesce che ha la capacità di cambiare sesso spontaneamente: si preveda quindi un metodo cambiaSesso() che provoca il cambio di sesso dell'oggetto; il sesso viene cambiato casualmente all'interno del metodo cresce()

SAPIENZA
UNIVERSITÀ DI ROMA
DIPARTIMENTO DI INFORMATICA

Metodologie di Programmazione

Lezione 19: Le interfacce

Esercizio: Successioni

- Progettare tre classi che realizzano diversi tipi di successioni
  - La successione { i^2 } (es. 0, 1, 4, 9, 16, ecc.)
  - La successione casuale (es. -42, 2, 5, 18, 154, ecc.)
  - La successione di Fibonacci (es. 1, 1, 2, 3, 5, 8, 13, ecc.)
- Elaborare un meccanismo generale per la generazione della successione indipendentemente dall'implementazione specifica

Esercizio: Animali

- Progettare una gerarchia di classi dei seguenti animali, ciascuno con determinate caratteristiche:
  - Uccello (vola, becca)
  - Pinguino (becca, nuota)
  - Aquila (vola, becca)
  - Pesce (nuota)
  - Pesce volante (nuota, vola)
  - Cane (salta, corre, fedele a, domestico)
  - Felino (salta, corre, fa le fusa)
  - Gatto (salta, corre, fa le fusa, domestico)
  - Uomo (salta, corre, pensa, nuota, vola?)

SAPIENZA
UNIVERSITÀ DI ROMA
DIPARTIMENTO DI INFORMATICA

Metodologie di Programmazione

Lezione 20: Le classi interne

Esercizio: Liste linkate di interi con classi interne

UNITELMA SAPIENZA

SAPIENZA
UNIVERSITÀ DI ROMA
DIPARTIMENTO DI INFORMATICA

- Implementare le liste linkate di interi in modo da nascondere all'esterno l'implementazione del singolo elemento della lista

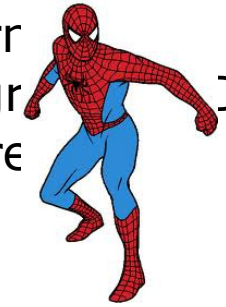
Esercizio:
Disney vs. Marvel (1)

UNITELMA SAPIENZA
SAPIENZA UNIVERSITÀ DI ROMA
DIPARTIMENTO DI INFORMATICA

- Modellare uno scontro su campo tra personaggi Disney e personaggi Marvel
- Squadra Disney = Paperinik, Ciccionik, Superpippo
  - Paperinik, Ciccionik e Superpippo sono la **version** **supereroe** dei ettivi per (Paperino, Pippo)



- Squadra Marvel = Spiderman, La cosa, Magneto
  - Notare che Spiderr **ve** **per** eter Parker, noto fotogr Da a co neto invece sono supere



Esercizio:
Disney vs. Marvel (2)

- I personaggi con una doppia vita devono esporre un'interfaccia DoppiaVita con i seguenti metodi:
  - assumiIdentitaSegreta(): consente di trasformarsi in supereroe
  - assumiIdentitaPubblica(): consente di assumere le sembianze 'umane'
- Ogni supereroe implementa l'interfaccia Supereroe con:
  - attacca(): sferra l'attacco specifico del supereroe
- Creare una partita su campo (ovvero una classe di test) in cui le due squadre si alternano in attacchi micidiali
  - Il nemico viene scelto casualmente dalla lista dei supereroi avversari

Esercizio:
Disney vs. Marvel (3)

UNITELMA SAPIENZA

SAPIENZA
UNIVERSITÀ DI ROMA
DIPARTIMENTO DI INFORMATICA

- Hint per versione **semplificata**: Senza utilizzare classi interne, ma solo interfacce ed ereditarietà
- Hint per versione **elaborata**: Per riflettere il fatto che nessuno è a conoscenza dei superpoteri in possesso dalle controparti umane, i supereroi, laddove possibile, dovrebbero estendere le classi umane corrispondenti ed essere implementati come loro classi annidate private (ad es. Paperinik estende Paperino, ma solo Paperino potrà restituire un'istanza di Paperinik tramite l'interfaccia DoppiaVita)

SAPIENZA
UNIVERSITÀ DI ROMA
DIPARTIMENTO DI INFORMATICA

Metodologie di Programmazione

Lezione 21: Le eccezioni

Esercizio:
Sequenza a gradini

- Definiamo una sequenza "a gradini" come una successione in cui ciascun elemento dista esattamente 1 dal precedente (la sequenza «7 8 9 10 11 12 11 10 11 10 9 8 7» è a gradini, mentre «1 2 8 7 6 5 42 9 20» non lo è)
- Costruire una sequenza a gradini inizialmente vuota
- Inserire successivamente altri elementi nella sequenza tramite il metodo aggiungi(int x)
- Nel caso il prossimo numero aggiunto violi il vincolo della sequenza a gradini, va notificato un **errore** al metodo chiamante
- Prevedere inoltre un metodo che stampi la sequenza finora memorizzata. La sequenza, inoltre, non ha vincoli di lunghezza (potenzialmente infinita)

Esercizio:
Floppy disk 2.5” (1)

UNITELMA SAPIENZA
SAPIENZA UNIVERSITÀ DI ROMA
DIPARTIMENTO DI INFORMATICA

- Implementare un floppy disk da 2.5 pollici
- Il floppy disk è un supporto magnetico che contiene dati e può essere acceduto in lettura e scrittura
- Esso ha una capacità pari a 1.474.560 bytes (ovvero 1.44 MB)
- Il floppy disk possiede anche un blocco scrittura che è possibile attivare o disattivare; questo meccanismo, se attivato, impedisce la scrittura di dati

Esercizio: Floppy disk 2.5" (2)

UNITELMA SAPIENZA
SAPIENZA UNIVERSITÀ DI ROMA
DIPARTIMENTO DI INFORMATICA

- Implementare inoltre i seguenti metodi:
  - posizionaTestina(): posiziona la testina in posizione k-esima
  - leggi(): legge i prossimi x byte
  - scrivi(): scrive i byte passati in input
  - formatta(): cancella tutti i dati presenti sul floppy disk
  - attivaBloccoScrittura(): attiva il blocco scrittura
  - disattivaBloccoScrittura(): disattiva il blocco scrittura
- Gestire inoltre tutte le situazioni di errore, ad esempio:
  - se si cerca di scrivere o formattare mentre è presente il blocco scrittura
  - se si cerca di scrivere ma il disco è pieno
  - si cerca di leggere ma non sono presenti sufficienti dati sul disco

Esercizio: Dizionario e Mappa

UNITELMA SAPIENZA — SAPIENZA UNIVERSITÀ DI ROMA, DIPARTIMENTO DI INFORMATICA

- Scrivere un'interfaccia **Dizionario** dotata dei metodi:
  - Elemento search(Chiave k): **cerca** l'elemento associato alla chiave k nella struttura dati
  - void add(Chiave k, Elemento e): **aggiunge** l'elemento e con chiave k nella struttura dati
  - Elemento delete(Chiave k): **rimuove** l'oggetto associato alla chiave k dalla struttura dati
  - int size(): restituisce la **taglia** del dizionario
- Implementare una coppia (k, e) chiave-elemento, costruita a partire da una Chiave k ed un Elemento e ad essa associato
- Implementare la struttura dati Mappa che rappresenta una collezione di coppie senza ripetizione di chiavi
- Prevedere il sollevamento delle seguenti eccezioni:
  - ElementNotFoundException: lanciata nel caso la chiave da cercare o rimuovere non è contenuta nella struttura dati

Esercizio:
Catena di volontari (1)

- Nel 1825 a causa del Miramichi Fire in Canada morirono 160 persone
- La squadra di volontari FIRE (Fuoco Immaginario o Reale Estinguesi) è specializzata nell'estinguere qualunque tipo di fuoco, sia esso fuoco fatuo o un devastante incendio naturale
- Dal gruppo emergono per competenza due figure:
  - La volontaria Acquafredda: la volontaria più veloce nel fornire secchi d'acqua al resto del gruppo
  - Il volontario Fuoco: il volontario dalla mira migliore e in grado di spegnere anche le fiamme più alte
- Il resto del gruppo è formato da semplici volontari
  uniti in una catena umana, ciascuno in grado di passare secchi d'acqua dal volontario alla sua sinistra a quello alla sua destra

Esercizio:
Catena di volontari (2)

UNITELMA SAPIENZA
SAPIENZA UNIVERSITÀ DI ROMA
DIPARTIMENTO DI INFORMATICA

- I volontari hanno a disposizione un **unico** secchio d'acqua che è possibile riempire o svuotare
- La volontaria Acquafredda è **l'unica** ad aver **accesso** all'acqua, ovvero l'unica in grado di riempire il secchio
- Il volontario Fuoco invece è **l'unico** in grado di osservare in maniera diretta l'incendio, l'unico a poterlo spegnere e l'unico anche a determinarne l'avvenuta estinzione
- All'incendio è associato un intero compreso tra 1 e 10 che ne determina l'intensità nella scala "Fireneit" e corrisponde, sperimentalmente, al numero di secchi necessari alla sua estinzion

Esercizio:
Catena di volontari (3)

UNITELMA SAPIENZA
SAPIENZA UNIVERSITÀ DI ROMA
DIPARTIMENTO DI INFORMATICA

- Quando l'incendio si estingue il volontario Fuoco si accorge dell'evento FuocoEstintoException e, secondo il codice condiviso all'interno della FIRE, lancia il messaggio 'BastaAcquaException!'
- Tale messaggio è molto importante ed è necessario che arrivi alle orecchie della volontaria Acquafredda il prima possibile
- Al ricevimento di tale messaggio, la volontaria Acquafredda dovrà smettere di riempire il secchio d'acqua, risorsa assai preziosa. Un obiettivo della missione dev'essere quello di non sprecare acqua ed utilizzarne la giusta quantità

Esercizio:
Catena di volontari (4)

UNITELMA SAPIENZA
SAPIENZA UNIVERSITÀ DI ROMA
DIPARTIMENTO DI INFORMATICA

- Costruire una catena di volontari avente in testa la volontaria Acquafredda, in coda il volontario Fuoco e in mezzo 3 volontari comuni
- I volontari avranno a disposizione un unico secchio, inizialmente messo a disposizione dalla volontaria Acquafredda, che viene passato da volontario a volontario tramite il metodo void estinguiIncendio(Secchio s)
- Appiccare un Incendio doloso e mettere in moto la catena FIRE in modo che estingua l'incendio

SAPIENZA
UNIVERSITÀ DI ROMA
DIPARTIMENTO DI INFORMATICA

Metodologie di Programmazione

Lezioni 22-23: Le collezioni

Esercizio:
il mio array iterabile

UNITELMA SAPIENZA
SAPIENZA
UNIVERSITÀ DI ROMA
DIPARTIMENTO DI INFORMATICA

- Scrivere una classe che implementa un array di interi iterabile
- Utilizzare l'interfaccia Iterable<Integer> e le classi interne

Esercizio: parentesi annidate

UNITELMA SAPIENZA
SAPIENZA UNIVERSITÀ DI ROMA
DIPARTIMENTO DI INFORMATICA

- Scrivere un metodo che, data una stringa in input, verifichi se le parentesi della stringa sono correttamente annidate
- Sono ammessi due tipi di parentesi: tonde e quadre

Esercizio: inversione degli elementi

- Scrivere un metodo che, data una coda in input, ne inverta gli elementi
- Scrivere la variante in cui viene fornita in input (e restituita in output) una stringa

Esercizio: calcolatrice in notazione polacca inversa

- Scrivere un metodo che, data una lista in input con operandi e operatori in notazione polacca inversa, calcoli il risultato delle operazioni
- Notazione polacca inversa: gli operatori vengono **DOPO** gli operandi (ad esempio: 3 4 + 5 - denota 3+4-5)

SAPIENZA
UNIVERSITÀ DI ROMA
DIPARTIMENTO DI INFORMATICA

Metodologie di Programmazione

Lezione 24: Input e output

Esercizio

- Scrivere una classe i cui oggetti rappresentano dei semplici dizionari di significati che memorizzano l'elenco dei significati per ciascuna parola del dizionario. Gli oggetti sono costruiti a partire da un oggetto di tipo File. Il file fornisce su ogni riga una coppia parola e significato. Ad esempio:
  - basso tenore
  - stella poligono
  - stella astro
  - stella VIP
  - basso chitarra
- La classe dispone dei seguenti metodi:
  - get che, data una parola, restituisce l'insieme dei significati della parola (ognuno rappresentato da una stringa).
  - toString che restituisce una rappresentazione sotto forma di stringa del dizionario. Tale rappresentazione mostra le parole in ordine lessicografico. Ad esempio, per il file mostrato sopra, restituisce prima tutti i significati di banco, poi quelli di basso, infine quelli di stella.

SAPIENZA
UNIVERSITÀ DI ROMA
DIPARTIMENTO DI INFORMATICA

Metodologie di Programmazione

Lezione 25: Ricorsione

Esercizio: Concatenazione ricorsiva di stringhe

- Si realizzi un metodo ricorsivo che, dato in input un array di stringhe, ne restituisca la loro concatenazione
- Ad esempio, concatena(new String[] { “abc”, “de”, “f” }) restituisce “abcdef”

Esercizio:
Ricerca binaria ricorsiva

UNITELMA SAPIENZA
SAPIENZA UNIVERSITÀ DI ROMA
DIPARTIMENTO DI INFORMATICA

- Scrivere un metodo ricorsivo per la ricerca binaria di un elemento all'interno di un array ordinato di interi
  - Restituendo -1 se l'elemento non è presente

Esercizio: Visualizzare il contenuto di una cartella

UNITELMA SAPIENZA
SAPIENZA UNIVERSITÀ DI ROMA
DIPARTIMENTO DI INFORMATICA

- Si realizzi una classe Cartella, costruita con il percorso di una cartella reale, dotata di un metodo toString() che ne visualizzi ricorsivamente il contenuto
- Avete bisogno della classe java.io.File

Esercizio: Cercare file in una cartella

UNITELMA SAPIENZA
SAPIENZA UNIVERSITÀ DI ROMA
DIPARTIMENTO DI INFORMATICA

- Si realizzi una classe Cartella, costruita con il percorso di una cartella reale, dotata dei seguenti metodi:
  - Cerca un file all'interno di una cartella
  - Cerca all'interno della cartella tutti i file con l'estensione specificata
  - Cerca all'interno della cartella tutti i file con le estensioni fornite in input
- Si faccia uso al meglio dell'overloading

Esercizio: Permutazioni di una stringa

UNITELMA SAPIENZA
SAPIENZA UNIVERSITÀ DI ROMA
DIPARTIMENTO DI INFORMATICA

- Scrivere una classe costruita con una stringa e dotata di un metodo generaPermutazioni che restituisce l'elenco completo delle permutazioni della stringa
  - ad esempio, per la stringa "abc" genera l'elenco ["abc", "acb", "bac", "bca", "cab", "cba"]
- Come procedere? E' necessario scomporre il problema in sottoproblemi
- Fissato un carattere della stringa iniziale, lo poniamo all'inizio della stringa e permutiamo la sottostringa rimanente
  - stesso problema, ma su una sequenza di caratteri più piccola!

Esercizio: Teseo, il Minotauro e il Labirinto di Creta

UNITELMA SAPIENZA

SAPIENZA
UNIVERSITÀ DI ROMA
DIPARTIMENTO DI INFORMATICA

- Si realizzi una classe Labirinto dotata di un solo ingresso a un corridoio
- Ciascun corridoio fornisce zero o più ingressi ad altri corridoi
- Ogni corridoio è dotato di un metodo contieneMinotauro che specifica se nel corridoio si trova il Minotauro
- La classe Labirinto è dotata di un metodo percorri() che, partendo dal corridoio iniziale, cerca il Minotauro nel labirinto
- **Avanzato:** Teseo, che fa il suo ingresso nel Labirinto, una volta trovato il Minotauro, stampa il percorso a ritroso mediante il "filo di Arianna"…

Variante: Versione con Labirinto a matrice

- Si realizzi una classe LabirintoMatrice in cui il labirinto è rappresentato mediante una matrice NxM (due valori in input al costruttore)
- Ogni cella della matrice è un'istanza di una classe Cella
  - Una cella o è una stanza o è un muro
  - Una stanza può o meno contenere il minotauro
- La classe LabirintoMatrice è dotata di un metodo percorri() che, partendo dalla cella (x,y) specificata in ingresso al metodo, cerca il Minotauro nel labirinto
- **Avanzato:** stampare il percorso a ritroso

SAPIENZA
UNIVERSITÀ DI ROMA
DIPARTIMENTO DI INFORMATICA

Metodologie di Programmazione

Lezione 26: Ricorsione (2); L'uguaglianza in Java

Esercizio: controlla numeri pari e dispari

UNITELMA SAPIENZA
SAPIENZA UNIVERSITÀ DI ROMA
DIPARTIMENTO DI INFORMATICA

- Scrivere una classe che, data una lista di interi, controlli ricorsivamente se ciascun numero in posizione pari sia dispari e ciascun numero in posizione dispari sia pari

Esercizio: Somma e sottrazione di valori

UNITELMA SAPIENZA
SAPIENZA UNIVERSITÀ DI ROMA
DIPARTIMENTO DI INFORMATICA

- Scrivere una classe SommaSottrai che, costruita con una lista di interi, espone un metodo ricorsivo sommaSottrai che restituisce la somma dei valori della lista in posizione dispari da cui vengono sottratti i valori in posizione pari
  - Ad esempio, data la lista [2, 5, 3, 7, 11, 1] il metodo restituisce 2-5+3-7+11-1=3

Esercizio: And tra espressioni in una lista

UNITELMA SAPIENZA
SAPIENZA UNIVERSITÀ DI ROMA
DIPARTIMENTO DI INFORMATICA

- Si progetti un'interfaccia Valutabile che espone un metodo valuta() il quale calcola il valore dell'oggetto e restituisce un booleano
- Si progetti quindi una classe ValutaValutabili i cui oggetti sono costruiti con un array di oggetti "valutabili"
- La classe espone un metodo valutaInAnd **ricorsivo di coda** il quale restituisce il valore booleano dell'and tra tutti i valori degli oggetti nell'array
- Analogamente, realizzare un metodo valutaInOr **ricorsivo di coda** che mette in or i valori degli oggetti nell'array

La Torre di Hanoi

- In un tempio dell'estremo oriente alcuni monaci devono spostare una pila di dischi d'oro da un piolo di diamante a un altro
- La pila iniziale comprende 64 dischi, tutti infilati nello stesso piolo dal basso verso l'alto in ordine di grandezza decrescente
- I monaci devono spostare l'intera pila su un altro piolo, rispettando il vincolo di muovere un solo disco per volta e non ponendo mai un disco più grande sopra uno più piccolo
- Sono disponibili 3 pioli e uno può essere usato come supporto temporaneo
- Si realizzi la classe TorreDiHanoi che risolva il problema

La Torre di Hanoi: suggerimenti

- Tenete bene a mente che per risolvere un problema ricorsivamente è necessario:
  1) ridurlo a un problema più piccolo
  2) identificare il caso base
- Posso riformulare il problema di spostare n dischi nel problema in cui:
  - Sposto n-1 dischi dal piolo 1 al piolo 2 (usando il piolo 3 come supporto temporaneo)
  - Sposto l'ultimo disco (il più grande) dal piolo 1 al piolo 3
  - Sposto n-1 dischi dal piolo 2 al piolo 3, usando il piolo 1 come supporto temporaneo

Esercizio: hashCode e equals con le collection

UNITELMA SAPIENZA
SAPIENZA UNIVERSITÀ DI ROMA
DIPARTIMENTO DI INFORMATICA

- Implementare una classe PunteggioGiocatore i cui oggetti memorizzano al loro interno l'informazione sul punteggio e sul nome del giocatore corrispondente
- Implementare correttamente i metodi hashCode e equals in modo da garantire la corretta implementazione dell'uguaglianza per la classe
- Verificare il comportamento della classe creando istanze di PunteggioGiocatore con gli stessi valori e con valori diversi
  - Aggiungendo riferimenti agli oggetti a una lista
  - Aggiungendo gli stessi riferimenti a un insieme

SAPIENZA
UNIVERSITÀ DI ROMA
DIPARTIMENTO DI INFORMATICA

Metodologie di Programmazione

Lezione 27: I tipi generici (parte 1)

Esercizio:
Lista Linkata generica

- Progettare una classe generica ListaLinkata con le principali operazioni disponibili nella classe omonima java.util.LinkedList

Esercizio: la classe Pila

- Progettare una classe generica Pila implementata mediante lista di elementi (provate anche con l'array)
- La classe è costruita con la dimensione iniziale dell'array ed implementa i seguenti metodi:
  - push: inserisce un elemento in cima alla pila
  - pop: elimina e restituisce l'elemento in cima alla pila
  - peek: restituisce l'elemento in cima alla pila
  - isEmpty: restituisce true se la pila è vuota

Esercizio:
Inverti lista e massimo

UNITELMA SAPIENZA
SAPIENZA UNIVERSITÀ DI ROMA
DIPARTIMENTO DI INFORMATICA

- Implementare i seguenti due metodi generici statici:
  - inverti, che data in input una lista di elementi di tipo generico, restituisca un'altra lista con gli elementi in ordine invertito
  - max, che data in input una lista di elementi di tipo generico, ne restituisca il valore massimo
- **Nota:** per il secondo metodo è necessario utilizzare il costrutto <T extends InterfacciaOClasse>, che impone un vincolo sul supertipo di T

SAPIENZA
UNIVERSITÀ DI ROMA
DIPARTIMENTO DI INFORMATICA

Metodologie di Programmazione

Lezione 28: I tipi generici (parte 2)

Esercizio: Successioni numeriche

UNITELMA SAPIENZA
SAPIENZA UNIVERSITÀ DI ROMA
DIPARTIMENTO DI INFORMATICA

- Scrivere una classe SuccessioniNumeriche, i cui oggetti sono inizialmente vuoti
- La classe espone un metodo addSuccessione che, preso in input un nome e una sequenza numerica, consente di aggiungere una successione (di lunghezza finita) del tipo numerico generico specificato (Integer, Float, Long, ecc.)
- La classe ha inoltre un metodo getSuccessione(String nome) con cui è possibile recuperare una successione a partire dal nome mnemonico
- Prevedere le seguenti successioni:
  - «Fibonacci»: 1, 1, 2, 3, 5, 8, 13, 21, 34
  - «1/n»: 1, 1/2, 1/3, 1/4, 1/5, 1/6
  - «RandomLong»: contiene oggetti Long costruiti casualmente
- **Hint:** utilizzare la classe Number come superclasse del tipo generico

Esercizio: cancellazione del tipo nella classe Pila

- Mostrare come viene cancellato il tipo dal compilatore nella classe Pila creata in un precedente esercizio
- Mostrare inoltre come viene trasformato il seguente codice:

```
Pila<Integer> p = new Pila<Integer>(10);
p.push(1);
p.push(2);
Integer k = p.pop();
```

Esercizio: MultiInsieme

- Definire una classe generica per gestire multiinsiemi, ovvero un insieme che può contenere più copie dello stesso elemento (ad esempio, { 1, 1, 2, 3, 3 }).
- Oltre a definire un costruttore che crea un multiinsieme vuoto, la classe deve definire i seguenti metodi:
  - Un metodo **add** che, dato un valore, lo aggiunge al multiinsieme, restituendo true se il valore non era già contenuto nel multiinsieme, false altrimenti.
  - Un metodo **get** che, preso in input un valore, restituisce il numero di copie di tale valore nel multinsieme (0 se il valore non è contenuto).
  - Un metodo **contains** che, preso in input un valore, restituisce true se il valore è contenuto nel multiinsieme, false altrimenti.
  - Un metodo **toSet** che restituisce l'insieme dei valori contenuti nel multiinsieme, ovvero senza duplicati
  - Un metodo **intersect** che, preso in input un multiinsieme set dello stesso tipo dell'oggetto su cui il metodo è invocato, modifica il multiinsieme di quest'ultimo in modo da contenere l'intersezione tra se stesso e set. Ad esempio, dato il multinsieme { 1, 1, 1, 2, 4, 4, 5 }, l'intersezione con il multinsieme { 1, 1, 2, 4, 7, 7 } modificherà il primo in { 1, 1, 2, 4 }

Esercizio: MultiMappa (1)

- Definire una classe generica per gestire multimappe. Una multimappa mantiene associazioni (chiave, insieme di valori) tali che ad ogni chiave è associato un insieme di valori. Il tipo delle chiavi può essere diverso da quello dei valori.
- Oltre a definire un costruttore che crea una multimappa vuota, la classe deve definire i seguenti metodi:
  - Un metodo **put** che, presa in input una chiave e un valore, aggiunge l'associazione alla multimappa, restituendo true se il valore non era già contenuto nell'insieme associato alla chiave, false altrimenti. Il metodo gestisce la situazione in cui la chiave specificata non esista, creando la nuova associazione.
  - Un metodo **get** che, presa in input una chiave, se la chiave è presente restituisce l'insieme ad essa associato, altrimenti restituisce null.

Esercizio: MultiMappa (2)

- Un metodo **contains** che, presa in input una chiave e un valore, restituisce true se l'associazione tra la chiave e il valore è contenuta nella multimappa, false altrimenti
- Un metodo **intersect** che, presa in input una chiave k e un insieme di valori set, se la chiave è presente rende l'insieme dei valori associato alla chiave uguale all'intersezione tra l'insieme originale e l'insieme set preso in input. Se set è pari a null, la chiave k viene rimossa dalla multimappa. Se la chiave non è presente, il metodo lancia l'eccezione IllegalArgumentException.
- Un metodo **intersectMultiMappa** che, presa in input una multimappa dello stesso tipo, rende la multimappa dell'oggetto su cui il metodo è invocato uguale all'intersezione delle due multimappe. In altre parole, rimarranno in questa multimappa solamente le chiavi che sono presenti anche nella multimappa presa in input e, per ognuna di queste chiavi, l'insieme dei valori diventerà uguale all'intersezione degli insiemi dei valori associati alla chiave nelle due multimappe.

SAPIENZA
UNIVERSITÀ DI ROMA
DIPARTIMENTO DI INFORMATICA

Metodologie di Programmazione

Lezione 29: Design pattern (parte 1)

Esercizio: Camminare

- Si considerino le seguenti cinque classi:

```
abstract public class EssereVivente
{
    abstract public void cammina();
}
```

```
public class Animale extends EssereVivente
{
    @Override
    public void cammina()
    {
        System.out.println("Sto camminando a 4 zampe...");
    }
}
```

```
public class Millepiedi extends Animale
{
    @Override
    public void cammina()
    {
        System.out.println("Sto camminando a 62 zampe...");
    }
}
```

```
public class Pianta extends EssereVivente
{
    @Override
    public void cammina()
    {
        System.out.println("Affonda le radici...");
    }
}
```

```
public class Uomo extends EssereVivente
{
    private int eta;

    @Override
    public void cammina()
    {
        if (eta >= 80) System.out.println("cammina con bastone");
        else if (eta >= 3) System.out.println("cammina su 2 arti");
        else System.out.println("cammina su 4 arti");
    }

    public int getEta() { return eta; }
    public void invecchia() { eta++; }
}
```

- Si mostrino le criticità e si ristrutturi il codice utilizzando il pattern Strategy

Esercizio: Array comparabile in modo flessibile

- Si progetti una classe generica che modella un array di elementi
- La classe, mediante lo strategy pattern, permette di modificare il modo di comparare gli array:
  - In ordine lessicografico (dal minore al maggiore)
  - In ordine lessicografico inverso (dal maggiore al minore)
  - Per dimensione degli array

SAPIENZA
UNIVERSITÀ DI ROMA
DIPARTIMENTO DI INFORMATICA

Metodologie di Programmazione

Lezione 30: Design pattern (parte 2)

Esercizio: Borsa (1)

- Si considerino le seguenti cinque classi:

```
public class Borsa
{
    private Analizzatore analizzatoreStatistiche;
    private Analizzatore analizzatoreTrend;

    // chiamato ogni volta che una nuova transazione è disponibile
    public void nuovaTransazione(Transazione t)
    {
        analizzatoreStatistiche.analizza(t);
        analizzatoreTrend.analizza(t);
    }
}
```

```
abstract public class Analizzatore
{
    abstract public void analizza(Transazione t);
}
```

```
public class AnalizzatoreStatistiche extends Analizzatore
{
    private List<Transazione> transazioni = new ArrayList<Transazione>();

    @Override
    public void analizza(Transazione t)
    {
        transazioni.add(t);
    }

    @Override
    public String toString()
    {
        double importo = 0.0;
        for (Transazione t : transazioni) importo += t.getImporto();
        return "IMPORTO MEDIO DELLE TRANSAZIONI = "+importo/transazioni.size();
    }
}
```

Esercizio: Borsa (2)


```
public class AnalizzatoreTrend extends Analizzatore
{
    private Transazione ultimaTransazione;
    private double trendImporto;

    @Override
    public void analizza(Transazione t)
    {
        if (ultimaTransazione == null) trendImporto = t.getImporto();
        else trendImporto = t.getImporto()-ultimaTransazione.getImporto();

        ultimaTransazione = t;
    }

    @Override
    public String toString()
    {
        return "TREND = "+trendImporto;
    }
}
```

```
public class Transazione
{
    private String compagnia;
    private double importo;

    public Transazione(String compagnia, double importo)
    {
        this.compagnia = compagnia;
        this.importo = importo;
    }

    public String getCompagnia() { return compagnia; }
    public double getImporto() { return importo; }
}
```

- Si mostrino le criticità e si ristrutturi il codice utilizzando il pattern Observer

Esercizio: Scrittori di libri (1)

- Si considerino le seguenti classi:

```
public class Scrittore
{
    public enum GenereLibro
    {
        AVVENTURA, GIALLO, FUMETTO
    }

    public Libro pubblica(GenereLibro genere)
    {
        Libro libro = null;

        switch(genere)
        {
            case AVVENTURA: libro = new LibroAvventura(); break;
            case GIALLO: libro = new LibroGiallo(); break;
            case FUMETTO: libro = new Fumetto(); break;
        }

        libro.impagina();

        return libro;
    }
}
```

```
public class ScrittoreDiGialli extends Scrittore
{

}
```

```
public class Fumettista extends Scrittore
{

}
```

Esercizio:
Scrittori di libri (2)

UNITELMA SAPIENZA
SAPIENZA
UNIVERSITÀ DI ROMA
DIPARTIMENTO DI INFORMATICA

```
abstract public class Libro
{
    public void impagina() { /* ... */ }
}

public class LibroGiallo extends Libro
{
}

public class LibroAvventura extends Libro
{
}

public class Fumetto extends Libro
{
}
```

- Si mostrino le criticità e si ristrutturi il codice utilizzando il pattern Factory

SAPIENZA
UNIVERSITÀ DI ROMA
DIPARTIMENTO DI INFORMATICA

Metodologie di Programmazione

Lezione 31: Design pattern (parte 3)

Esercizio

- Modificare la seguente classe utilizzando un design pattern appropriato in modo tale che la stessa, unica configurazione sia accessibile da qualsiasi classe del progetto

```
public class Configuration
{
    public final static String FILENAME = "config/project.properties";
    private Map<String, String> config = new HashMap<String, String>();

    public Configuration()
    {
        BufferedReader br = null;

        try
        {
            br = new BufferedReader(new FileReader(FILENAME));
            while(br.ready())
            {
                String[] pair = br.readLine().split("=");
                config.put(pair[0], pair[1]);
            }
        }
        catch(IOException e)
        {
            e.printStackTrace();
        }

        // ...
    }

    public String get(String name) { return config.get(name); }
}
```

SAPIENZA
UNIVERSITÀ DI ROMA
DIPARTIMENTO DI INFORMATICA

Metodologie di Programmazione

Lezione 32: Espressioni lambda

Esercizio: array con ordinamento dinamico in Java 8

- Svolgere nuovamente l'esercizio sull'array con ordinamento dinamico utilizzando il paradigma funzionale per implementare lo strategy pattern
  - Utilizzare anche il metodo di default reversed dell'interfaccia Comparator (disponibile in Java 8)

Esercizio

- Scrivere un'espressione lambda che, data in input una stringa, la divida utilizzando lo spazio come carattere separatore e restituisca una lista ordinata delle sottostringhe risultanti

SAPIENZA
UNIVERSITÀ DI ROMA
DIPARTIMENTO DI INFORMATICA

Metodologie di Programmazione

Lezione 33: La reflection

Esercizio: GestoreClasse

- Implementare una classe GestoreClasse i cui oggetti sono costruiti a partire dal nome di una classe sotto forma di stringa
- La classe implementa i seguenti metodi:
  - getNomiCampi che restituisce un insieme ordinato dei nomi dei campi della classe in questione
  - getNomiMetodi che restituisce un insieme ordinato dei nomi dei metodi della classe in questione
  - getNomiMetodi che, dato in input il tipo di ritorno, restituisce l'elenco dei metodi aventi tale tipo di ritorno
  - invoca che, dato il nome di un metodo sotto forma di stringa, l'oggetto su cui eseguire il metodo e l'elenco dei parametri del metodo, invoca il metodo con i parametri corrispondenti

Esercizio: MiniInterprete

- Implementare una classe generica MiniInterprete i cui oggetti sono costruiti a partire da un oggetto del tipo generico specificato
- La classe espone il metodo parse che, data una stringa nel formato nomeMetodo(param1, …, paramN), invoca il corrispondente metodo con N parametri
- Si assuma che i tipi dei parametri siano scelti tra: Integer, Boolean, String e Double
- Si assuma che non esista più di un metodo con lo stesso nome e N parametri
- Si gestiscano le eccezioni in modo appropriato
- Ad esempio:

  new MiniInterprete("ciao").parse("length()") restituisce 4

  new MiniInterprete("ciao").parse("charAt(3)") restituisce 'o'