ESAME 16 SETTEMBRE 2021

1. (4 punti) Descrivere il meccanismo dell'overloading e overriding. Produrre un esempio **minimale** scritto in Java che concretizzi quanto spiegato.

L'OVERLOADING E L'OVERRIDING SONO DUE MECCANISMI LEGATI AL CONCETTO DI POLIMORFISMO IN JAVA,
SI PARLA DI OVERLOADING QUANDO, ALL'INTERNO DI UNA STESSA CLASSE, SI DEFINISCONO PIU' METODI CON LO STESSO NOME E TIPO DI VALORE DI RITORNO, MA CHE HANNO DIVERSI TIPI E QUANTITA' DI PARAMETRI IN ENTRATA.

esempio di overloading:

```
PUBLIC CLASS INTCASTER {
    PUBLIC INT CAST (DOUBLE VALUE){
        RETURN (INT) VALUE; }
    PUBLIC INT CAST(FLOAT VALUE){
        RETURN (INT) VALVE;}
}
```

L'OVERRIDING, DESCRIVE LA POSSIBILITA' DI DEFINIRE UN METODO IN UNA CLASSE, GIA' DEFINITO NELLA SUA SUPERCLASSE, COSI' DA SOVRASCRIVERLO E RENDERLO PIU' SPECIFICO ED ADATTO ALLE FUNZIONALITA' DI TALE SOTTOCLASSE.

esempio di overriding:

```
PUBLIC CLASS SQUARE{
    INT BASE;
    INT HEIGHT;
    PUBLIC VOID PRINTINFO(){
        SYSTEM.OUT.PRINTLN(BASE);
        SYSTEM.OUT.PRINTLN(HEIGHT);
    }
}
```

```
PUBLIC CLASS CSQUARE EXTENDS SQUARE{
    FLOAT COLOR;
    @OVERRIDE
    PUBLIC VOID PRINTINFO(){
        SYSTEM.OUT.PRINTLN(SUPER.BASE);
        SYSTEM.OUT.PRINTLN(SUPER.HEIGHT);
        SYSTEM.OUT.PRINTLN(COLOR);}
}
```

2. (2 punti) Per ogni costrutto iterativo, indicare il numero di volte per il quale viene eseguito il suo corpo. Se non diversamente indicato, si assume che la variabile contatore non venga modificata all'interno del corpo di ciascun costrutto iterativo.

(a) `for(int i = 8; i >= -3; i = i + 2){...}` INFINITO

(b) `for(int i = 0; i < 10; i--){...}` INFINITO

(c) `for(int i = 10; i > 0; i++){...}` INFINITO

(d) `for(int i = -10; i >= 0; i--){...}` 0

(e) `for(int i = -2; i >= 0; i--){...}` 0

(f) `for(int i = -8; i <= 3; i = i - 4){...}` INFINITO

(g)
```
for (int k = 0; k < 20; k+=2) { 3
if (k - 3 == 1) break;
}
```

3. Si vuole realizzare un programma in Java che rispecchi il "Gioco delle sedie musicali".

```
IMPORT JAV.UTIL.*;

PUBLIC CLASS PLAYER{
    PRIVATE INT COD;
    PRIVATE STRING NAME;
    PRIVATE INT AGE;
    PUBLIC PLAYER(INT C,STRING N,INT A){
        COD = C;
        NAME = N;
        AGE = A;}
    PUBLIC INT GETCOD(){ RETURN COD;}
    PUBLIC INT GETAGE(){ RETURN AGE;}
    PUBLIC STRING GETNAME(){ RETURN NAME;}
}
PUBLIC CLASS GAMEMANAGER{
    ARRAYLIST<PLAYER> PLAYERS=NEW ARRAYLIST<PLAYER>();
    INT AVAILABLESEAT;
    PUBLIC GAMEMANAGER(STRING PATH){
        FILE F=NEW FILE(PATH);
        TRY{
            SCANNER S=NEW SCANNER(F);
            WHILE(S.HASNEXTLINE()){
                STRING LINE[]=S.NEXTLINE.SPLIT(";");
                PLAYER TMPPLAYER=NEW PLAYER(LINE[0],LINE[1],LINE[2]);
                PLAYERS.ADD(TMPPLAYER);
            SYSTEM.OUT.PRINTLN("IL GIOCO INIZIA CON"+PLAYERLIST.SIZE()+" GIOCATORI");
            AVAILABLESEAT=PLAYERLIST.SIZE()-1;
        } CATCH (FILENOTFOUNDEXCEPTION){ SYSTEM.OUT.PRINTLN("ERRORE FILE");}
    }
    PUBLIC VOID REMOVEPLAYER(){
        RANDOM RNG=NEW RANDOM();
        PLAYER CHOOSEN=PLAYERLIST.REMOVE(RNG.NEXTINT(PLAYERLIST.SIZE()));
        AVAILABLESEAT-=1;
        SYSTEM.OUT.PRINTLN(CHOOSEN.GETNAME()+" È STATO ELIMINATO");
    }
    PUBLIC VOID ROUND(){
        RANDOM RNG=NEW RANDOM();
        INT MILLISECONDS=(RNG.NEXTINT(20)+1)*1000;
        SYSTEM.OUT.PRINTLN("LA MUSICA INIZIA...");
        THREAD.SLEEP(MILLISECONDS);
        REMOVEPLAYER();
    }
    PUBLIC BOOLEAN FINISHED(){
        RETURN PLAYERLIST.SIZE()==1;
    }
    PUBLIC VOID WIN(){
        SYSTEM.OUT.PRINTLN("HA VINTO :"+PLAYERLIST.GET(0).GETNAME());
    }
}
PUBLIC STATIC VOID MAIN(STRING[] ARGS){
    STRING PATH = "D:/PERCORSO/FILE.TXT"; // IL FILE CON I GIOCATORI
    GAMEMANAGER GAME=NEW GAMEMANAGER(PATH);
    WHILE(!GAME.FINISHED()){
        GAME.ROUND();
    }
    GAME.WIN();
}
```

4. Realizzare un programma Java il quale, dato un file di testo:

   (a) (2 punti) Legga il suo contenuto

   (b) (4 punti) Crei un secondo file di testo con la lista delle parole uniche, senza ripetizioni

   (c) (2 punti) Stampi a video il numero di parole presenti nel file di input e il numero di parole uniche trovate

NB: la scelta della struttura dati più opportuna è una parte fondamentale ai fini della valutazione dell'esercizio. Si assume inoltre che le parole siano sempre e solo divise da uno spazio.

```
PUBLIC STATIC VOID MAIN(STRING[] ARGS){
        FILE F=NEW FILE("PERCORSO/FILE.TXT");
        INT TOTALWORDS=0;
        TRY{
                SCANNER S=NEW SCANNER(F);
                PRINTWRITER FILEW =NEW PRINTWRITER("PERCORSO/FILENUOVO.TXT");
                TREESET<STRING> UNIQUEWORDS= NEW TREESET<STRING>();
                WHILE(S.HASNEXT()){ UNIQUEWORDS.ADD(S.NEXT()); TOTALWORDS++;}
                ITERATOR IT = UNIQUEWORDS.ITERATOR();
                WHILE(IT.HANNEXT()){FILEW.PRINTLN(IT.NEXT());}
        }CATCH(FILENOTFOUNDEXCEPTION E){SYSTEM.OUT.PRINTLN("FILE NON TROVATO");}
        SYSTEM.OUT.PRINTLN("PAROLE TOTALI:"+TOTALWORDS+" PAROLE UNICHE:"+UNIQUEWORDS.SIZE());
}
```