ESAME 11 GIUGNO 2023

1. Descrivere il meccanismo dell'ereditarietà in modo esaustivo spiegando il suo funzionamento e come i modificatori di visibilità si relazionano a tale meccanismo. Produrre un esempio (con codice minimale) che concretizzi quanto richiesto.

L'ereditarietà, è una tecnica di programmazione legata alla programmazione ad oggetti; consiste nel modellare e raggruppare classi secondo delle relazioni di di "estensione", ossia definire delle classi comuni più generali, facendo sì che più classi che dovrebbero modellare gli stessi comportamenti, possano comunemente essere estensione della stessa classe padre, "ereditando" appunto da essi, metodi ed attributi, aggiungendone poi di propri più specifici; ad esempio, le due classi "UOMO" e "CANE" saranno estensione della stessa classe "MAMMIFERO", ma solo la classe "UOMO" avrà l'attributo specifico "IDIOMA", comunemente, la classe padre si dice SUPERCLASSE, la classe figlio: SOTTOCLASSE. una sottoclasse di base, può ereditare metodi e attributi della sua superclasse, a meno che non siano PRIVATE.

2. Spiegare le principali differenze tra l'utilizzo di un array e della classe *ArrayList* nella gestione di una sequenza di elementi.

nonostante il nome sia simile, esse sono due strutture dati diverse, l'array è un insieme STATICO di soli dati di tipo primitivo, una volta definita la sua dimensione, essa non sarà mutabile, inoltre tutti gli elementi di un array devono essere dello stesso tipo, un array list è una struttura dati più complessa, estende la classe LINKEDLIST e descrive una collezione di oggetti di dimensione variabile, può contenere elementi di tipo diverso, anche non primitivi, fra utilizzo di GENERICS è quindi possibile definire un array list di un tipo specifico, se non specificato, tutti gli elementi verranno trattati (e ritornati) come di tipo OBJECT, sarà quindi necessaria una conversione di dato esplicita se si volessero fare operazioni specifiche.

3. 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 = 2; i <= 9; i++){...}` 8

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

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

   (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){...}` 3

   (g)
   ```
   for (int k = 0; k < 20; k+=2) { 10
      if (k + 3 == 1) System.out.println(k + " ");
      }
   ```

4. Realizzare la classe *Customer* al fine di gestire le informazioni relative ad un cliente di una campagna di promozione commerciale. Tale campagna si articola in questo modo: dopo aver effettuato una serie di acquisti per almeno $100, il cliente riceverà uno sconto di $10 sull'acquisto successivo. Progettare quindi i seguenti metodi:

```
public void makePurchase(double amount) // registra un acquisto
public boolean discountReached() // true se e solo se l'utente ha raggiunto lo sconto
```

Al fine di verificare la correttezza delle propria implementazione, si realizzi un programma di collaudo (test). Esso dovrà rappresentare la seguente situazione: un cliente dovrà ottenere uno sconto, il quale lo utilizzerà per fare degli acquisti. L'importo totale derivante da tali acquisti dovrà essere maggiore di $90 ma minore di $100 in modo tale da non poter ottenere uno sconto. Successivamente, l'utente dovrà effettuare un altro acquisto, il quale lo porterà a godere dello sconto sull'acquisto successivo.

classe customer

```
IMPORT JAVA.UTIL.*;
PUBLIC CLASS CUSTOMER{
        INT DISCOUNTS;
        DOUBLE SPENTMONEY;
        PUBLIC CUSTOMER(INT D){
                DISCOUNTS = D;
        }
        PUBLIC VOID MAKEPURCHASE(DOUBLE AMOUNT){
                WHILE(DISCOUNTS >0){
                        DISCOUNTS--;
                        AMOUNT -=10;
                }
                SPENTMONEY += AMOUNT;
                WHILE(DISCOUNTREACHED()){
                        DISCOUNTS+=1;
                        SPENTMONEY-=100.;
                }
        PUBLIC BOOLEAN DISCOUNTREACHED(){ RETURN SPENTMONEY>=100;}
        PUBLIC VOID GETDISCOUNTS(){ RETURN DISCOUNTS;}
}
```

classe di collaudo

```
IMPORT JAVA.IO.*;
PUBLIC CLASS ESTESTER{
        PUBLIC VOID RUN(){
                SCANNER IN=NEW SCANNER();
                DOUBLE INPUT=0;
                CUSTOMER C=NEW CUSTOMER();
                WHILE(INPUT>=0){
                        SYSTEM.OUT.PRINTLN("INSERT AMOUNT (NEG TO EXIT)");
                        INPUT=IN.NEXTDOUBLE();
                        IF(IN>0){ C.MAKEPURCHASE(IN);}
                }
                SYSTEM.OUT.PRINTLN("REMAINS DISCOUNTS: "+C.GETDISCOUNTS);
```

ES. 4

```
PUBLIC CLASS ACCOUNT {
    PRIVATE INT COD;
    PRIVATE STRING NAME;
    PRIVATE STRING SURNAME;
    PRIVATE FLOAT AMOUNT;
    PUBLIC ACCOUNT( INT C, STRING N, STRING S, FLOAT A) {
        COD = C;
        NAME = N;
        SURNAME = S;
        AMOUNT = A;
    }
    PUBLIC ACCOUNT(STRING LINE){
        STRING[] RAW = LINE.SPLIT( );
        COD = RAW [0];
        NAME = RAW[1];
        SURNAME = RAW [2];
        AMOUNT = RAW[3];
    }
    PUBLIC   INT    GETCOD     { RETURN COD ;}
    PUBLIC  STRING  GETNAME    { RETURN NAME;}
    PUBLIC  STRING  GETSURNAME { RETURN SURNAME ;}
    PUBLIC  FLOAT   GETAMOUNT  { RETURN AMOUNT;}
}
```

main

```
IMPORT JAVA.UTIL.*;
IMPORT JAVA.IO.*;
PUBLIC CLASS MAIN {
    PUBLIC STATIC VOID MAIN (STRING[] ARGS) {
        FILE F=NEW FILE("<PERCORSO>");
        ARRAYLIST<ACCOUNT> ACCOUNTS = NEW ARRAYLIST<ACCOUNT>();
        TRY {
            SCANNER IN = NEW SCANNER(F);
            WHILE(IN.HASNEXT()){ ACCOUNTS.ADD(NEW ACCOUNT(IN.NEXTLINE()));}
        } CATCH(FILENOTFOUNDEXCEPTION E){
            SYSTEM.OUT.PRINTLN("ERRORE FILE");
            RETURN;
        }
        ACCOUNT RICHEST =(ACCOUNT) ACCOUNTS.GET(0);
        FLOAT AVG;
        FLOAT N;
        SYSTEM.OUT.PRINTLN("I CONTI ROSSI SONO:");
        FOR(INT i=0; i<ACCOUNTS.SIZE(); i++){
            ACCOUNT TMP = (ACCOUNT)ACCOUNTS.GET(i);
            IF(TMP.GETAMOUNT()< 0){ SYSTEM.OUT.PRINTLN(TMP.GETNAME()+" "+TMP.GETSURNAME());}
            ELSE {
                N += 1.0;
                AVG += TMP.GETAMOUNT();
            }
            IF( TMP.GETAMOUNT() > RICHEST.GETAMOUNT()){
                RICHEST = TMP;
            }
        }
        SYSTEM.OUT.PRINTLN("IL SALDO MEDIO E'"+STRING.VALUEOF(AVG/N));
        SYSTEM.OUT.PRINTLN("IL PIU' RICCO E'"+RICHEST.GETNAME()+" "+RICHEST.GETSURNAME());
    }
}
```