REPORT CYCLONE

SIMONE SUDATI

Docente: Angelo Gargantini

1045936

—

Ingegneria Informatica

—

Università di Bergamo

INTRODUZIONE

Il linguaggio C presenta diverse situazioni in cui l'uso dei puntatori è cosiddetto "unsafe". In particolare con: puntatori nulli, assegnamento esplicito ad un puntatore e aritmentica dei puntatori. Cyclone permette di gestire queste casistiche rispettivamente con: controllo dereferenziazione di puntatore nullo, impedendo cast da int a puntatore e impedendo l'aritmetica dei puntatori fuori dai bounds.
Vedremo come da un programma scritto in linguaggio C si possa, attraverso delle modifiche, riscriverlo in Cyclone per renderelo "safe".

INDICE

PROGRAMMA

Il programma è stato scritto da zero in linguaggio C per assicurarsi che funzionasse correttamente e senza alcun errore né in fase di compilazione né in fase di esecuzione. A partire da esso sono state attuate le modifiche necessarie per scriverlo in Cyclone e renderlo safe.

Il programma riguarda la stima di funzioni. In particolare, dati dei valori campionari ricevuti in input in un sistema, esso consente una prima stima della distribuzione della funzione. Permetterà quindi di capirne e analizzarne l'andamento.

Ipotesi: Sono disponibili n valori x_i indipendenti e identicamente distribuiti con media μ e varianza σ^2

Problema (classico): Stimare μ e σ^2

Media campionaria:

$$\bar{X} = \frac{\sum_{i=1}^{n} x_i}{n}$$

Varianza campionaria:

$$S^2 = \frac{\sum_{i=1}^{n} (x_i - \bar{X})^2}{n-1}$$

Per ottenere ciò, si inizia con una stima della deviazione standard che indica di quanto si discostano in media i valori dal valore atteso (valore medio). La stima della deviazione standard la si ottiene tramite lo stimatore deviazione standard campionario. (la deviazione stadard è la radice quadrata della varianza).
Poi si stima il valore atteso, attraverso lo stimatore media campionaria e infine si ricava il range all'interno del campione, che è un importante indice di dispersione, calcolandone massimo e minimo valore campionario .

Una volta determinati questi parametri si può passare allo studio della distribuzione della funzione. In questo programma verranno utilizzate le seguenti distribuzioni:

- Distribuzione Normale
- Distribuzione di Poisson (o esponenziale negativa)
- Distribuzione Esponenziale (o exponent)
- Distribuzione Erlagiana(o Erlang)
- Distribuzione Logistica (o logistic)

E attraverso i parametri calcolati precedentemente (Deviazione standard, valore atteso, range ecc.) inseriti nelle relative distribuzioni, potremo capire che andamento seguano i dati in input in termini di andamento della distribuzione di probabilità.

INPUT e OUTPUT

Il programma richiede all'utente quanti valori desidera inserire e poi successivamente richiede di inserire uno alla volta i valori di cui dispone.
Da questi valori il programma procederà al calcolo della deviazione standard, della media campionaria(stima del valore atteso), del minino e massimo campionario e del range.
Ad esempio:

```
BENVENUTO NELLA STIMA DI FUNZIONI DI SIMONE

Stima della deviazione standard.
Enter the numbers of elements: 5
1. Enter number: 2.5
2. Enter number: 3.2
3. Enter number: 4.7
4. Enter number: 6.1
5. Enter number: 4.3

Standard Deviation= 1.392121

Stima della media.
Enter the numbers of elements: 5
1. Enter number: 2.5
2. Enter number: 3.2
3. Enter number: 4.7
4. Enter number: 6.1
5. Enter number: 4.3

Media= 4.160

Stima del range.
Enter the numbers of elements: 5
1. Enter Number : 2.5
2. Enter Number : 3.2
3. Enter Number : 4.7
4. Enter Number : 6.1
5. Enter Number : 4.3

Range= 3.600
```

Successivamente il programma richiede all'utente di inserire il nome della funzione di distribuzione dei dati. (Il sistema riconosce la stringa indipendentemente dalla notazione in maiuscolo o minuscolo utilizzata). Riconoscera la distribuzione "Normale" in tutte le seguenti casistiche che possono presentarsi : "NORMALE","normale", "NorMALE", "noRmAlE" ecc.
Ecco i seguenti esempi:

Inserendo in input come distribuzione di funzione "NORMALE"

```
--------------------------------------------------
RECAP
Funzione: normale --> changeCase
Distribuzione: ( 1/sqrt(2piGreco*1.89^2) )*exp ^-(x-4.10)^2/(2*1.89^2)
Standard Deviation= 1.891
Media campionaria= 4.100
Min campionario= 3.100
Max campionario= 6.900
Range= 3.800
--------------------------------------------------
```

Inserendo in input come distribuzione di funzione “poisson”

```
-----------------------------------------------------
RECAP
Funzione: POISSON --> changeCase
Distribuzione: ( exp-^(2.00)*2.00^X ) / ( X! )
Standard Deviation= 2.003
Media campionaria= 3.740
Min campionario= 2.700
Max campionario= 6.300
Range= 3.600
-----------------------------------------------------
```

Inserendo in input come distribuzione di funzione “ERlang”

```
-----------------------------------------------------
RECAP
Funzione: erLANG --> changeCase
Distribuzione: ( 0.35*X^(k-1) )/ (k-1)!  con X>=0
Standard Deviation= 2.887
Media campionaria= 4.340
Min campionario= 1.700
Max campionario= 8.900
Range= 7.200
-----------------------------------------------------
```

Inserendo in input come distribuzione di funzione “ExpOnEnt”

```
-----------------------------------------------------
RECAP
Funzione: eXPoNeNT --> changeCase
Distribuzione: 0.22*exp^-0.22*X
Standard Deviation= 2.201
Media campionaria= 4.540
Min campionario= 3.200
Max campionario= 7.200
Range= 4.000
-----------------------------------------------------
```

Inserendo in input come distribuzione di funzione “logISTIC”

```
-----------------------------------------------------
RECAP
Funzione: LOGistic --> changeCase
Distribuzione: ( e^-(x-3.60)/(s) ) / ( s*(1+ e^-(x-3.60)/(s)))
Standard Deviation= 2.720
Media campionaria= 3.600
Min campionario= 1.500
Max campionario= 8.300
Range= 6.800
-----------------------------------------------------
```

Invece, inserendo un nome differente da quelli presenti nel sistema, il programma si limiterà a indicare che è una “distribuzione sconosciuta”.

CARATTERISTICHE DI CYCLONE

L'obbiettivo è riuscire a prevenire i seguenti casi:

1) NULL pointers
   Quando ha una dereferenziazione cyclone inserisce un controllo di non nullità prima dell'accesso al valore puntato.

2) Buffer overflow
   Per prevenire buffer overflows, cyclone limita l'aritmetica dei puntatori a solo determinati tipi di puntatori che vedremo in seguito.

3) zero terminated strings
   Quando fai aritmetica x+i, cyclone inserisce un controllo che non ci sia carattere null in x.

4) Bounded pointers
   Puntatori dotati di informazione aggiuntiva circa la dimensione dell'array in memoria a cui puntano. Questo rende possibile per il compilatore fare dei bounds checking.

5) dangling pointers
   Un puntatore che punta ad una zona di memoria che è stata deallocata (ad esempio una zona dello stack che viene liberata da un pop). La soluzione sta nell'usare la malloc.

Le modifiche introdotte per passare dal codice scritto in C al codice scritto in cyclone sono le seguenti:

- Puntatori non nulli @notnull (o brevemente @): per evitare di far svolgere dei controlli inutili quando si sa già che il puntatore è non nullo.
- Puntatori Fat @fat(o brevemente con ?): mantengono un'informazione agguntiva sulla dimensione dell'array a cui si può accedere con la funzione numelts(). Unici puntatori con cui è permessa l'aritmetica dei puntatori.
- Stringhe char*@zeroterm: per indicare le stringhe che sono sicuramente terminate dal carattere di fine stringa. Bisogna fare attenzione ad usarle perché può diventare dispendioso.

PER ESEGUIRE DA TERMINALE DI CYGWIN PROGRAMMA C

1) Cambiare la directory: *cd /*
2) Individuare programma desiderato: *ls*
3) Compilare il programma: *cyclone -o NomeExe NomeProgramma.c*
4) Lanciare l'exe: */NomeExe.exe*

PER ESEGUIRE DA TERMINALE DI CYGWIN PROGRAMMA CYC

1) Cambiare la directory: *cd /*
2) Individuare programma desiderato: *ls*
3) Compilare il programma: *cyclone -o NomeExe NomeProgramma.cyc*
4) Lanciare l'exe: */NomeExe.exe*

REPORT CPP

SIMONE SUDATI

Docente: Angelo Gargantini

1045936

—

Ingegneria Informatica

—

Università di Bergamo

INTRODUZIONE

In questo progetto scritto in linguaggio C++ verranno utilizzati diversi costrutti tipici di questo linguaggio (ereditarietà a diamante, ereditarietà privata, metodi virtual, enumerativi, distruttori, inizializer list ecc.). Tutti utilizzati in uno scenario concreto che è la gestione dei dolci, in particolare la ricerca dei dolci in un ricettario in cui sono ordinati e distinti secondo un codice identificativo.

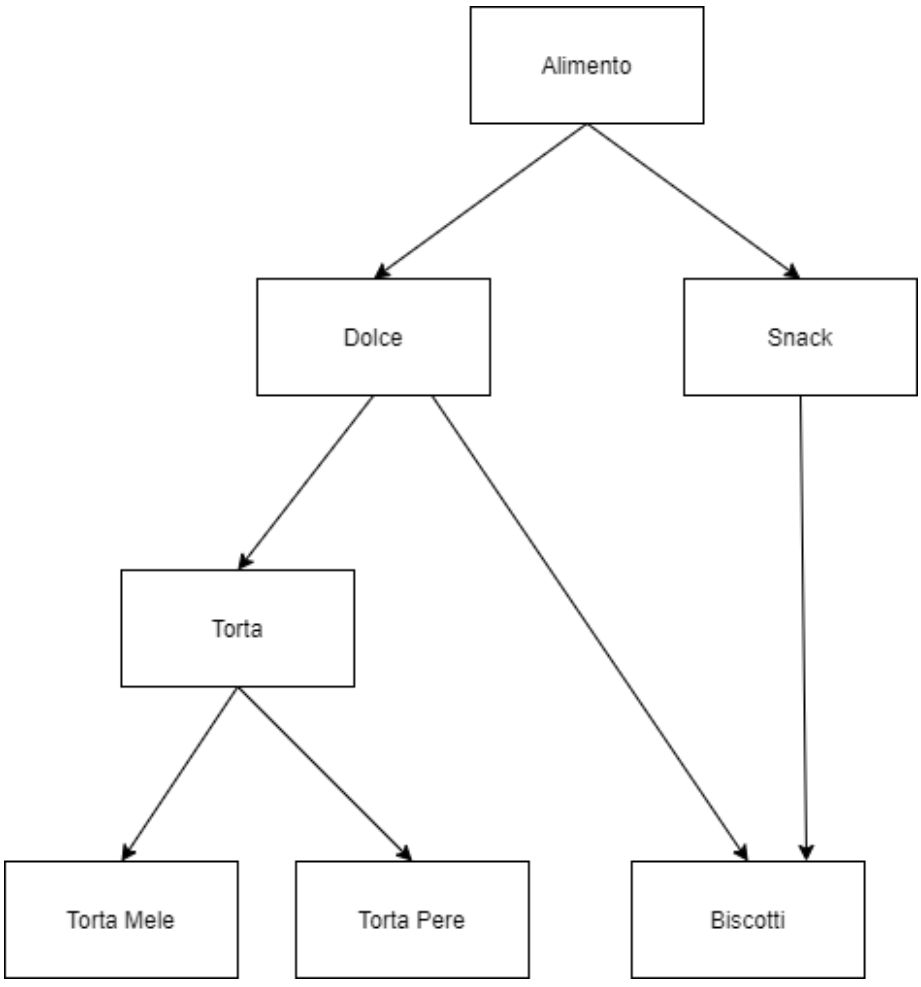
INDICE

PROGRAMMA

Il programma si occupa della gestione dei dolci all'interno di un ricettario.
Ogni dolce è ordinato e identificato da un codice identificativo.
Il programma ha l'obiettivo di creare un output ordinato a forma di ricettario in cui ogni dolce appena inserito contenga tutti i propri parametri che lo caratterizzano. Tutti i dolci inoltre hanno un determinato apporto calorico calcolato con metodi particolari a seconda della tipologia di dolce in questione.

STRUTTURA PROGRAMMA

La classe astratta Alimento (una classe astratta è una classe in cui è dichiarato almeno un metodo virtuale puro) è la classe padre da cui ereditano le classi figlie Dolce e Snack che dovranno implimentare i metodi della classe astratta. Dolce a sua volta è classe padre della classe Torta, che è padre delle classi figlie TortaMele e TortaPere; invece la classe Biscotti è soggetta a ereditarietà multipla. Notiamo la struttura ereditaria a diamante in cui Biscotti eredita da due classi che hanno il medisimo padre. (Sia Dolce che Snack hanno l'ereditarietà "segnata virtual" dal padre Alimento).
Il programma si compone quindi di 7 classi e del file di main.



CODICE DEI COSTRUTTI UTILIZZATI

Nel programma sono stati utlizzati i seguenti costrutti:

A) Costruttore con Default inizialization

```
1.  Dolce::Dolce(string nome,int codRicetta=0,int grammiZucchero=0,bool freddo=false){
2.    this->nome=nome;
3.    this->codRicetta=codRicetta;
4.    this->grammiZucchero=grammiZucchero;
5.    this->freddo=freddo;
6.  }
```

B) Costruttore con Inizialization list

```
1.  Alimento::Alimento(string nome):nome(nome){
2.    cout<<"CA ";
3.  }
```

```
1.  Torta(int minForno,int numUova,char*descr):minutiForno(minForno),numeroUova(numUova),descrizione(descr){
2.    cout<<"inizializer list";
3.  }
```

C) Metodo Pure virtual

```
1.  virtual int CalcolaCalorie()=0;
```

D) Ereditarietà privata

```
1.  class Snack: private virtual Alimento{
2.  public:
3.    enum Orario { colazione=0, merenda=1, pastoMezzanotte=2 };
4.    string sceltaOrario;
5.    Snack();
6.    Snack(string nome);
7.    Snack(string nome,enum Orario time);
8.    ~Snack(){};
9.    void stampa();
10.   int CalcolaCalorie();
11. };
```

Quindi eredita il campo nome da Alimento come privato. Il campo era pubblico ma viene ereditato privatamente quindi si può accedere solamente con metodi in Alimento.

E) Ereditarietà multipla (con virtual per l'eredità a diamante)

Della classe biscotti che eredità da snack e dolce che hanno il medesimo padre Alimento in cui i metodi sono definiti virtual ed ereditano con ereditarietà "segnata come virtual" (snack e dolce da Alimento).

F) Overriding (di metodi definiti virtual nel padre)

```
1.  void Dolce::stampa(){
2.     Alimento::stampa();
3.     cout<<" è un DOLCE."<<endl;
4.     cout<<"Codice ricetta: "<<Dolce::codRicetta<<endl;
5.     cout<<"Freddo: "<<Dolce::freddo<<endl;
6.     cout<<"Grammi zucchero: "<<Dolce::grammiZucchero<<endl;
7.  }
```

G) Enumerativi

```
1.  enum Orario {
2.    colazione=0,
3.    merenda=1,
4.    pastoMezzanotte=2
5.  };
```

OUTPUT PROGRAMMA

```
Notazione costruttori:
 CA= costruttore Alimento
 CD= costruttore Dolce
 CT= costruttore Torta
 CTM= costruttore TortaMele
 CTP= costruttore TortaPere
 CS= costruttore Snack
 CB= costruttore Biscotti

---------------------------------
CA CD
Strudel è un DOLCE.
Codice ricetta: 1
Freddo: 1
Grammi zucchero: 13
Calcola calorie: -metodo di dolce- 52
---------------------------------
CA CD CT
Crostata è un DOLCE.
Codice ricetta: 2
Freddo: 1
Grammi zucchero: 2
E'una TORTA.
Minuti forno: 20
Numero uova: 3
Calcola calorie: -metodo di torta- 6
---------------------------------
CA CD CT CTM
Crostata mele è un DOLCE.
Codice ricetta: 3
Freddo: 1
Grammi zucchero: 2
E'una TORTA.
Minuti forno: 30
Numero uova: 5
E' una TORTA DI MELE.
Numero mele: 3
Calcola calorie: -metodo di tortaMele- 6
---------------------------------
CA CD CT CTP
Crostata pere è un DOLCE.
Codice ricetta: 4
Freddo: 1
Grammi zucchero: 2
E'una TORTA.
Minuti forno: 28
Numero uova: 7
E' una TORTA DI PERE.
Numero pere: 5
Calcola calorie: -metodo di tortaPere- 10
---------------------------------
```

```
---------------------------------
CA CS
Ovetto Kinder è un SNACK.
Periodo giornata: merenda
Calcola calorie: -metodo di Snack- 2
---------------------------------
CA CD CS CB
Le Gocciole è un DOLCE.
Codice ricetta: 5
Freddo: 1
Grammi zucchero: 12
Le Gocciole è un SNACK.
Periodo giornata: pastoMezzanotte
BISCOTTO di forma goccia
Calcola calorie: -metodo di biscotti- 24
---------------------------------
```

REPORT ASMETA

SIMONE SUDATI

Docente: Angelo Gargantini

1045936

—

Ingegneria Informatica

—

Università di Bergamo

INTRODUZIONE

Di seguito verrà implementata una piccola applicazione in Asmeta. Innanzitutto verrà rappresentata la macchina a stati finiti dell'applicazione e poi la si svilupperà in codice. Con uso dei costrutti di AsmetaL come domini, funzioni, regole di transizione e assiomi. Infine verrà anche aggiunto un caso d'uso di utilizzo dell'applicazione.

INDICE

PROGRAMMA

Si vuole modellare la gestione dei referti medici online per i pazienti di un ospedale.
Il cliente potrà accedere tramite le proprie credenziali inserendo prima il proprio ID Utente e in seguito la password.

→ Se non risultano presenti nel database (o Id Utente rifiutato o Password rifiutata) il sistema richiede nuovamente l'inserimento delle credenziali.
Gli ID Utente presenti nel sistema attualmente sono:

```
domain UserIDList = { 1045936, 123423, 4562456, 5644665}
```

ai quali corrispondono rispettivamente le seguenti password:

```
”Miao1”, ”Miao2”, “Miao3”, “Miao4”.
```

→ Se entrambe le credenziali sono state inserite correttamente, il cliente si troverà nel menu principale, in cui potrà scegliere tra diverse funzioni:

1) Sezione referti (poter inserire il codice di un referto per poterlo scaricare come pdf).

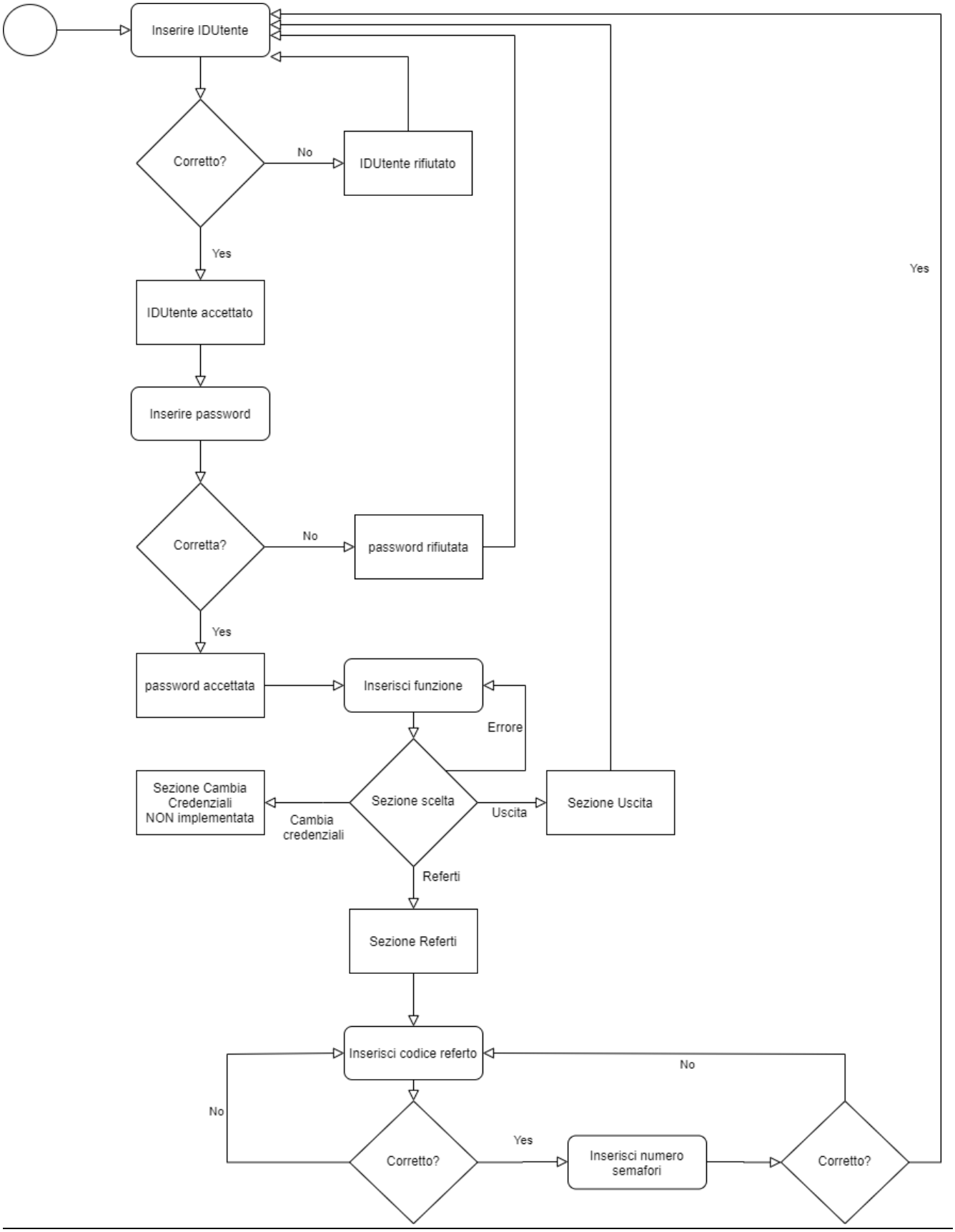
   Finchè il codice del referto inserito risulta assente nel database viene richiesto di inserire il codice corretto del referto. Una volta inserito il codice corretto del referto bisogna anche sostenere una prova di non essere un robot, contando il numero di semafori presenti nell'immagine (=3). Un inserimento errato del numero di semafori riporta all'inserimento del codice del referto invece se si risponde correttamente si ottiene il referto come pdf e si esce dal sistema.
   Attualmente nel sistema sono presenti i seguenti referti:

   ```
   domain Referti = { 1000 , 1500 , 2006 , 2020 , 1818}
   ```

2) Sezione cambio credenziali (per poter cambiare la propria Pws e il proprio ID)
   NON implementata, per possibli implementazioni future.

3) Sezione Uscita (per effettuare il logout).

RAPPRESENTAZIONE MACCHINA A STATI FINITI



ESEMPIO DI UN CASO D'USO : FILE.TEST

```
load SPEC_NAME.asm
check currentState = START;
step
check currentState = INSERIREID;
check messaggio = Inserire ID Utente;
set inputInteger := 1045936;
step
check currentState = CHECKID;
check messaggio = Inserire ID Utente;
check currentUserID = 1045936;
set inputInteger := 1045936;
step
check currentState = INSERIREPWD;
check messaggio = Inserire la Password;
check currentUserID = 1045936;
set inputInteger := 1045936;
set inputString := "miao1";
step
check currentState = CHECKPWD;
check messaggio = Inserire la Password;
check currentUserID = 1045936;
check currentPwd = miao1;
set inputInteger := 1045936;
set inputString := "miao1";
step
check currentState = CHOOSESERVICE;
check messaggio = Di che servizio vuoi usufruire? REFERTI o CHANGEPWS o EXIT;
check currentUserID = 1045936;
check currentPwd = miao1;
set inputInteger := 1045936;
set inputString := "miao1";
set selectedService := REFERTI;
step
check currentState = INSERIRERFT;
check messaggio = Inserisci codice del referto;
check currentUserID = 1045936;
check currentPwd = miao1;
set inputInteger := 2006;
set inputString := "miao1";
set selectedService := REFERTI;
step
check currentState = CHECKRFT;
check messaggio = Inserisci codice del referto;
check currentUserID = 1045936;
check currentPwd = miao1;
check currentRFT = 2006;
set inputInteger := 2006;
set inputString := "miao1";
set selectedService := REFERTI;
step
check currentState = CHECKNOROBOT;
check messaggio = Quanti semafori ci sono nell'immagine?;
check currentUserID = 1045936;
check currentPwd = miao1;
```

```
check currentRFT = 2006;
set inputInteger := 2;
set inputString := "miao1";
set selectedService := REFERTI;
step
check currentState = INSERIRERFT;
check messaggio =  Reinserisci il codice del referto.. puoi essere un robot! ☹;
check currentUserID = 1045936;
check currentPwd = miao1;
check currentRFT = 2006;
set inputInteger := 2006;
set inputString := "miao1";
set selectedService := REFERTI;
step
check currentState = CHECKRFT;
check messaggio =  Reinserisci il codice del referto.. puoi essere un robot! ☹;
check currentUserID = 1045936;
check currentPwd = miao1;
check currentRFT = 2006;
set inputInteger := 2006;
set inputString := "miao1";
set selectedService := REFERTI;
step
check currentState = CHECKNOROBOT;
check messaggio = Quanti semafori ci sono nell'immagine?;
check currentUserID = 1045936;
check currentPwd = miao1;
check currentRFT = 2006;
set inputInteger := 3;
set inputString := "miao1";
set selectedService := REFERTI;
step
check currentState = START;
check messaggio =  Eccoti il referto ☺;
check currentUserID = 1045936;
check currentPwd = miao1;
check currentRFT = 2006;
step
```

ESEMPIO DI UN CASO D'USO : ANIMATOR

State 0	State 1	State 2	State 3	State 4	State 5
START	INSERIREID	CHECKID	INSERIREPWD	CHECKPWD	CHOOSESERVICE
	Inserire ID Utente	Inserire ID Utente	Inserire la Password	Inserire la Password	Di che servizio vuoi usufruire? REFERTI o CHANGEPWS o EXIT
	1045936	1045936	1045936	1045936	1045936
		1045936	1045936	1045936	1045936
			"miao1"	"miao1"	"miao1"
				miao1	miao1
					REFERTI

State 6	State 7	State 8	State 9
INSERIRERFT	CHECKRFT	CHECKNOROBOT	START
Inserisci codice del referto	Inserisci codice del referto	Quanti semafori ci sono nell'immagine?	Eccoti il pdf del referto :)
1000	1000	3	3
1045936	1045936	1045936	1045936
"miao1"	"miao1"	"miao1"	"miao1"
miao1	miao1	miao1	miao1
REFERTI	REFERTI	REFERTI	REFERTI
	1000	1000	1000

CODICE

```
asm Referto_Medico_Online

import StandardLibrary

signature:
    domain UserIDList subsetof Integer
    domain Referti subsetof Integer
    domain Semafori subsetof Integer
    enum domain State = {START | INSERIREID | CHECKID | INSERIREPWD | CHECKPWD | CHOOSESERVICE| INSERIRERFT | CHECKRFT |CHECKNOROBOT }
    enum domain Service = {REFERTI | CHANGEPWS | EXIT}
    dynamic controlled noRobot: Referti -> Integer
    dynamic controlled currentState : State
    dynamic controlled messaggio : String
    dynamic controlled currentUserID : Integer
    dynamic controlled currentPwd : String
    dynamic controlled currentRFT : Integer
    dynamic monitored inputString : String
    dynamic monitored inputInteger : Integer
    dynamic monitored selectedService: Service
    static isValidID: Integer -> Boolean
    static isValidPwd: String -> Boolean
    static isValidRFT: Integer -> Boolean
    static isValidNumSemafori: Integer->Boolean


definitions:
    // User ID registrati nel sistema
    domain UserIDList = { 1045936 , 123423 , 4562456 , 5644665}
    // Referti presenti nel sistema
    domain Referti = { 1000 , 1500 , 2006 , 2020 , 1818}
    // Numero di semafori presenti nell'immagine (per confermare di non essere un robot)
    domain Semafori= {3}


    function isValidID($id in Integer) =
        if(exist $u in UserIDList with $u = $id ) then
            true
        else
            false
        endif

    function isValidPwd($pwd in String) =
        if (($pwd ="miao1"and currentUserID=1045936) or ($pwd ="miao2"and currentUserID=123423) or ($pwd ="miao3"and currentUserID=4562456) or ($pwd ="miao4"and currentUserID=5644665) ) then
            true
        else
            false
        endif


    function isValidRFT($rft in Integer) =
        if(exist $r in Referti with $r = $rft ) then
            true
        else
            false
        endif

    function isValidNumSemafori($sem in Integer) =
        if(exist $s in Semafori with $s = $sem ) then
            true
```

```
        else
            false
        endif


    macro rule r_insertID =
        if(currentState = INSERIREID) then
            par
                currentUserID := inputInteger
                currentState := CHECKID
            endpar
        endif

    macro rule r_checkID =
        if(currentState = CHECKID) then
                if(isValidID(currentUserID)) then
                    par
                        currentState := INSERIREPWD
                        messaggio := "Inserire la Password"
                    endpar
                else
                    par
                        currentState := START
                        messaggio := "ID Utente inesistente!"
                    endpar
                endif
        endif

    macro rule r_insertPWD =
        if(currentState = INSERIREPWD) then
            par
                currentPwd := inputString
                currentState := CHECKPWD
            endpar
        endif

    macro rule r_checkPWD =
        if(currentState = CHECKPWD) then
            if(isValidPwd(currentPwd)) then
                par
                    currentState := CHOOSESERVICE
                    messaggio := "Di che servizio vuoi usufruire? REFERTI o CHANGEPWS o EXIT"
                endpar
            else
                par
                    currentState := START
                    messaggio := "Password errata!"
                endpar
            endif
        endif


    macro rule r_chooseService =
        if(currentState=CHOOSESERVICE) then
            par
                if(selectedService=REFERTI) then
                    par
                        currentState := INSERIRERFT
                        messaggio := "Inserisci codice del referto"
                    endpar
                endif

                if(selectedService=EXIT) then
                    par
```

```
                        currentState := START
                        messaggio := "Uscita dal sistema!"
                    endpar
                endif
                if(selectedService=CHANGEPWS) then
                    par
                        //TODO: per implementazioni future!
                        currentState:= START
                        messaggio := "Uscita dal sistema!"
                    endpar
                endif
            endpar
        endif

    macro rule r_insertRFT =
        if(currentState = INSERIRERFT) then
            par
                currentRFT := inputInteger
                currentState := CHECKRFT
            endpar
        endif

    macro rule r_checkRFT =
        if(currentState = CHECKRFT) then
                if(isValidRFT(currentRFT)) then
                    par
                        currentState := CHECKNOROBOT
                        messaggio := "Quanti semafori ci sono nell'immagine?"
                    endpar
                else
                    par
                        currentState := INSERIRERFT
                        messaggio := "Referto inesistente!"
                    endpar
                endif
        endif

    macro rule r_checkNOROBOT =
        if(currentState = CHECKNOROBOT) then
            if( isValidNumSemafori(inputInteger)) then
                par
                    messaggio :=" Eccoti il pdf del referto :) "
                    currentState := START
                endpar
            else
                par
                    messaggio :=" Reinserisci il codice del referto.. puoi essere un robot! :( "
                    currentState := INSERIRERFT
                endpar
            endif
        endif


    main rule r_Main =
        if(currentState = START) then
            par
                currentState := INSERIREID
                messaggio := "Inserire ID Utente"
            endpar
        else
            par
                r_insertID[]
                r_checkID[]
                r_insertPWD[]
```

```
188.                 r_checkPWD[]
189.                 r_chooseService[]
190.                 r_insertRFT[]
191.                 r_checkRFT[]
192.                 r_checkNOROBOT[]
193.             endpar
194.         endif
195.
196.
197.default init s0:
198.     function currentState = START
```

REPORT SCALA

SIMONE SUDATI

Docente: Angelo Gargantini

1045936

—

Ingegneria Informatica

—

Università di Bergamo

INTRODUZIONE

Di seguito verrà implementata una piccola applicazione Object Oriented in linguaggio Scala.
Verrà implementato un trait, una classe astratta e diverse altre classi. Verranno anche usati metodi che utilizzano costrutti quali filter, map, foldLeft ecc.
L'ambito è quello dei dolci e si vuole calcolare le calorie nei vari alimenti per capirne in quale quantità se ne posso mangiarne per rimanere al di sotto di una determinata soglia di calorie.

INDICE

PROGRAMMA

L’ambito è culinario e in particolare quello dei dolci.
L’obbiettivo è calcolare le calorie nei dolci (che possono essere torte o biscotti) per capire in quale quantità possono essere mangiati per rimanere al di sotto di una determinata soglia di calorie acquisite.
Ogni ingrediente (ad esempio uova, farina e zucchero) ha una determinato apporto calorico misurato come calorie al grammo e verrà di conseguenza classificato in:

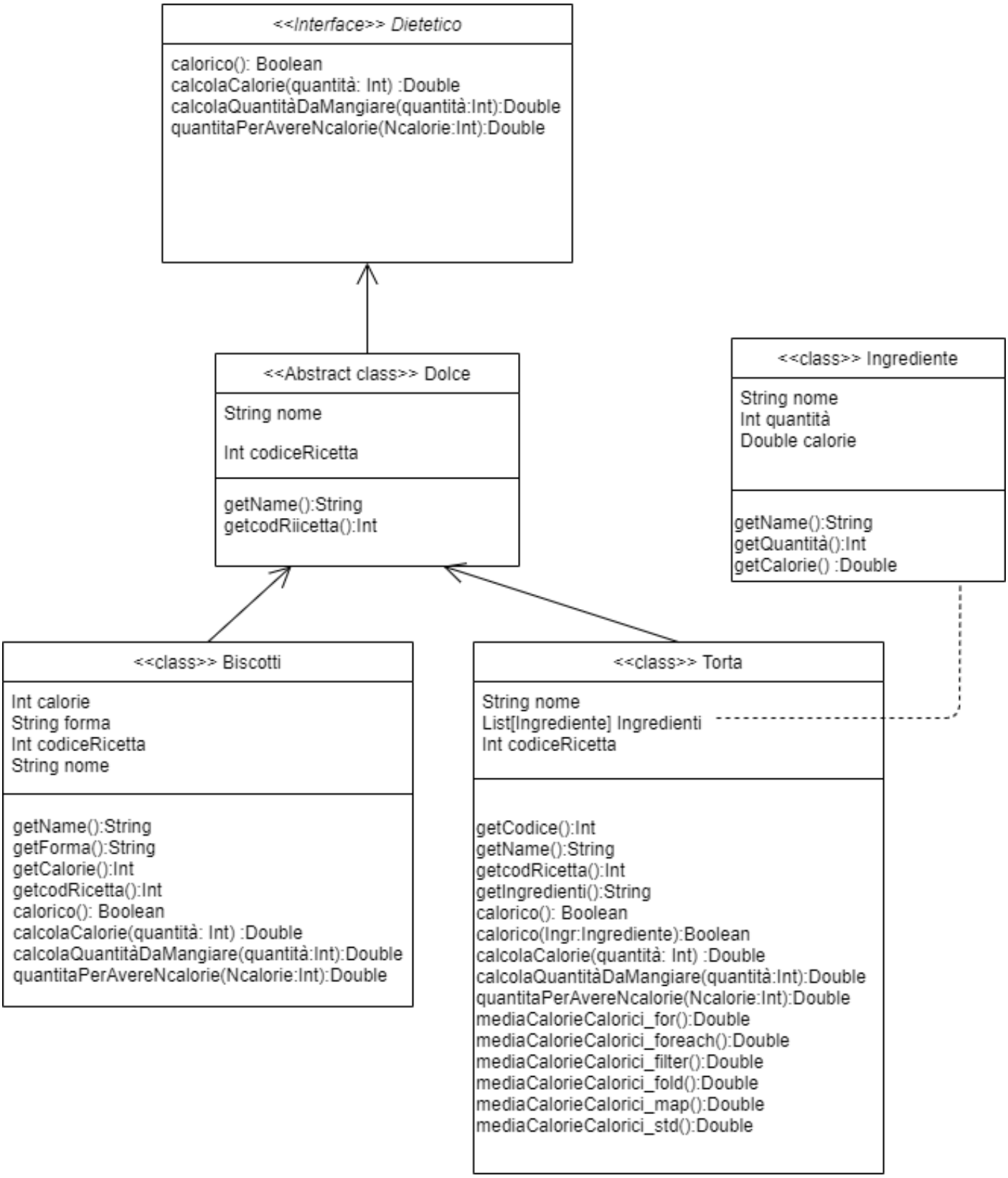
→ Ingrediente calorico se contiene > 100 calorie al grammo
→ Ingrediente non calorico se contiene <= 100 calorie al grammo

Creata la torta con i relativi ingredienti utilizzati per prepararla (con quantitativi e apporti calorici), il programma calcola la quantità in grammi che se ne può mangiare come quantità massima assumibile.
E in seguito all’inserimento come input della quantità richiesta di torta da mangiare, il programma accosente se la quantità selezionata è minore dell’apporto calorico consentito altrimenti indica la massima quantità assumibile.
Esempio: la quantità di calorie massima assumibile ad ora nel programma è impostata a 500 calorie. Se la richiesta di quantitativo di torta non supera tale soglia allora viene accettata la richiesta, altrimenti vengono calcolati i grammi necessari per assumere massimo 500 calorie dalla torta.

Oltre alle torte poi vengono calcolate le calorie nei boscotti.
Diversamente dalle torte i biscotti non hanno ingredienti calorici/non calorici ma sono loro stessi a essere distinti in calorici/non calorici. Infatti per i biscotti diversamente dalle torte si può leggere sulla confezione la quantità di calorie nel singolo biscotto.
Allo stesso modo delle torte, si può procedere alla richiesta di poter mangiare una determinata quantità di biscotti che può essere accettata se è sotto la soglia (nei biscotti fissata nel programma a 200 calorie) oppure rifiutata proponendo la nuova quantità da assumere calcolata in modo da assumere al massimo 200 calorie di biscotti.

CLASS DIAGRAM

La struttura del programma è rappresentata dal seguente class diagram.



STRUTTURA DEL PROGRAMMA

E' presente un'interfaccia (trait) Dietetico che contiene 4 metodi astratti che dovranno essere implementati dalle classi che la implementano. L'interfaccia a sua volta viene implementata da una classe astratta Dolce che contiene due campi che sono il nome del dolce e il codice nel ricettario del dolce. La classe astratta viene estesa da due classi Torta e Biscotti. Inoltre la classe Torta presenta come campo una lista di Ingredienti definiti nella classe Ingrediente.

Sia Torta che Biscotti implementano i metodi di Dietetico:

- Calorico() : metodo per calcolare se un determinato ingrediente/dolce supera una determinata soglia di calorie. Se la dovesse superare viene ritornato il valore true altrimenti false.

- CalcolaCalorie(quantità) : metodo che si occupa di calcolare il totale delle calorie di una determinata quantità di dolce.

- CalcolaQuantitaDaMangiare(quantità) : in base all'apporto calorico che porterebbe mangiare quel dolce questo metodo calcola quanti grammi è possibile mangiare del dolce in questione. Se la quantità inserita come input rispetta le calorie viene accettata altrimenti viene ritornato il valore precedentemente calcolato.

- QuantitàPerAvereNcalorie(Ncalorie) : metodo che restituisce la quantità in grammi del dolce in questione necessari per totalizzare le calorie richieste.

ESEMPIO OUTPUT

```
----------------------------------------------------------------------

Torta: Crostata
Codice: 8
Ingredienti :Farina Zucchero Uova

Farina calorico? true
Zucchero calorico? false
Uova calorico? True

Quantità che posso mangiarne: 1.0 grammi
Quantità che posso mangiarne: 6.150061500615006 grammi
Quantità che posso mangiarne: 6.150061500615006 grammi

Media calorici -for-: 81.3
Media calorici -foreach-: 81.3
Media calorici -filter-: 81.3
Media calorici -fold-: 81.3
Media calorici -map-: 81.3
Media calorici -std-: 81.3
Media calorici -curr-: 81.3

----------------------------------------------------------------------

Biscotto: Le Gocciole
Calorico? false
Calorie: 61.0

Quantità che posso mangiarne: 30.0 grammi
Quantità che posso mangiarne: 33.0 grammi
Quantità che posso mangiarne: 33.0 grammi

----------------------------------------------------------------------
```

COMMENTO

Notiamo come sia nei dolci che nei biscotti abbiamo inserito in input tre richieste di quantitativi da poter mangiare. Inizialmente nei dolci abbiamo richiesto di poter mangiare 1 grammo e la richiesta viene accettata perché il programma calcola che si rimane sotto la soglia calorica. Viceversa per le successive due richieste, in cui il quantitativo richiesto supera la soglia calorica allora il programma fissa il quantitativo alla quantità massima che si può assumere per non superare la soglia calorica (fissata a 6.15 grammi per la torta).
Stesso ragionamento per i biscotti in cui la quantità massima assumibile è 33 grammi, eventualità che si verifica nella seconda e terza richiesta dove erano entrambe maggiori di 33 grammi. Invece la richiesta di poterne mangiare 30 grammi (la prima richiesta) viene accettata.