Progetto di Sistemi di Comunicazione Multimediali

Relazione di progetto

Giovanni Dini - gioggi2002@gmail.com - g.dini3@campus.uniurb.it
Matricola 232274 • Università degli Studi di Urbino "Carlo Bo" • Sessione Autunnale 2012

Indice

Analisi dei requisiti

OBIETTIVO:

Si predisponga un algoritmo per l'implementazione dell'algoritmo Deficit Round Robin (DRR) sviluppato in qualsiasi linguaggio e qualsiasi ambiente. La relazione deve dimostrare il funzionamento dell'algoritmo.

REQUISITI:

L'algoritmo Deficit Round Robin è una versione modificata dell'algoritmo Weighted Round Robin. Non necessita di conoscere a priori la dimensione del pacchetto che dovrà essere servito ed è quindi ideale per gestire un traffico di pacchetti di dimensione variabile e non prevedibile, in quanto la sua complessità è O(1).

FUNZIONAMENTO:

Come funziona l'algoritmo DRR?

- Serviamo ad ogni coda un quanto di bits (può essere un valore unico o variabile per code diverse)
- Prendiamo il primo pacchetto in attesa della coda e analizziamo la sua dimensione confrontandola con il quanto attuale*:

  - se la dimensione è minore (o uguale) del quanto, serviamo il pacchetto e settiamo il quanto attuale alla dimensione del pacchetto meno il quanto attuale

  - se la dimensione è maggiore del quanto, non serviamo il pacchetto e incrementiamo il contatore del deficit di un altro quanto

- Procediamo con un'altra coda

* quanto attuale: quanto di base + deficit accumulato

Progettazione concettuale

SCELTE DI PROGETTO:

Per familiarità e rapidità di sviluppo, il progetto è stato sviluppato in Java, seguendo ovviamente il paradigma di programmazione ad oggetti.

Il programma genera diverse code, le quali generano a loro volta diversi pacchetti (al momento è impostato un numero casuale limitato tra 1 e 10, ma è fornita la possibilità di generarne un numero predefinito semplicemente decommentando una parte di codice già fornita).

Degna di nota la scelta di progetto di utilizzare come elemento chiave per la sincronizzazione il costrutto *ReentrantLock*. Quali vantaggi porta questa scelta? Il *ReentrantLock* accetta nel costruttore un parametro opzionale: la "fairness". Settando questo parametro a *false* o lasciandolo di default il comportamento del lock è che, una volta rilasciato dal thread che lo aveva acquisito, ogni altro thread avrà la possibilità di acquisirlo, in ordine del tutto imprevedibile (in parole povere: non c'è un ordine stabilito o una priorità). Se invece impostiamo il parametro fairness a *true* il risultato sarà che il processo in attesa di acquisirle il lock da più tempo sarà il primo ad ottenerlo.
Visto che ogni thread acquisisce il lock, esegue e successivamente lo rilascia, otterremo in questo modo la garanzia di un ordine ben definito e rispettato, che ricalca fedelmente il meccanismo round robin.

Per mantenere inoltre una vaga condizione di realismo, si è deciso di non sincronizzare la partenza delle code, simulando così l'arrivo di pacchetti in una coda a tempi casuali (in sostanza, non tutte le code sono generate nello stesso momento - può capitare che la prima coda effettui qualche round prima ancora che le altre abbiano terminato di generare i propri pacchetti o che la coda numero 1 non sia la prima a partire).

ENTITÀ COINVOLTE (CLASSI):

- Main
- Coda
- Pacchetto

ANALISI DELLE CLASSI:

Main:

La main stampa i messaggi iniziali di interazione con l'utente e genera le code.

Inoltre contiene il lock necessario per assicurare l'ordine delle operazioni delle code.

Una volta finita la generazione delle code, termina.

Coda (id):

La coda è la classe più importante del progetto. Riceve come unico parametro in ingresso il suo id (per renderla più riconoscibile quando dovrà stampare messaggi).

Il suo funzionamento si struttura in questa maniera:

- Genera una lista (coda) in cui inserire i pacchetti.
- Genera una quantità random di pacchetti tramite l'apposito metodo.
- Fin quando la lista (coda) non sarà stata svuotata, esegue il metodo service() per servire i pacchetti in attesa.

Metodi della classe coda:

ArrayList generaPacchetti(coda, numero di pacchetti):

Il metodo generaPacchetti genera (come prevedibile) i pacchetti che saranno messi in coda. Il numero di pacchetti è generato casualmente (infatti al momento il secondo parametro è inutilizzato). Inoltre, una volta generato il pacchetto, lo inserisce nella coda che gli viene fornita come parametro. Terminata la generazione dei pacchetti e il loro inserimento, restituisce la lista (coda) contenente tutti i pacchetti da servire.

int generaDimensione():

Questo metodo viene richiamato dalla funzione generaPacchetti per generare la dimensione casuale del pacchetto. Non accetta parametri e restituisce la dimensione generata.

void disegno(coda, deficit risultato, deficit iniziale, id della coda):

Il metodo di disegno si occupa di stampare a schermo lo stato della coda. I parametri richiesti sono la coda (per prendere i valori - in realtà questo parametro non sarebbe necessario, ma è stato incluso per comodità, nel caso in cui si volessero stampare più valori dalla coda, per esempio per mostrare anche il pacchetto seguente), il deficit risultante dall'operazione di servizio, il deficit prima dell'operazione di servizio e l'id della coda (per chiarezza di stampa).

void service(coda):

Il metodo service è il cuore del nostro algoritmo. Acquisisce il lock e controlla se il pacchetto può essere servito o meno. Se il pacchetto viene servito, lo elimina dalla coda (calcolando il giusto deficit), altrimenti esegue le operazioni di aumento del deficit. Infine, rilascia il lock.
Il parametro preso in ingresso è la coda da cui prendere o rimuovere i pacchetti.
Questo metodo, inoltre, richiama il metodo disegno() per stampare a video i messaggi che spiegano l'avanzamento dell'algoritmo.

Pacchetto(dimensione):

Questa classe serve solamente a generare l'oggetto che rappresenta il pacchetto. Il suo unico parametro e variabile interna è la dimensione dello stesso.

Codice sorgente

Nota: il codice sorgente è riportato male indentato a causa delle impostazioni del documento.

MAIN.JAVA:

```
package progettoscm1213;

import java.util.Scanner;
import java.util.concurrent.locks.ReentrantLock;

/**
 *
 * @author Giovanni Dini
 */
public class Main {
    public static int quantum;
    public static int npacchetti;
    public static final ReentrantLock queue = new
ReentrantLock(true);

    /**
     * @param args the command line arguments
     */
    public static void main(String[] args) throws
InterruptedException
    {

        System.out.println("/
****************************************/");
        System.out.println("* PROGRAMMA DI SIMULAZIONE DI UN
ALGORITMO *");
        System.out.println("*           DEFICIT ROUND ROBIN
*");
        System.out.println("/
****************************************/");
```

```
        System.out.println("*    PROGETTO DI SISTEMI DI COMUNICAZIONE
*");
        System.out.println("*       MULTIMEDIALI - A.A. 2012/2013
*");
        System.out.println("/
****************************************/");
        System.out.println("*    GIOVANNI DINI - MATRICOLA 232274
*");
        System.out.println("/
****************************************/");
        System.out.println("\n");
        System.out.println("Il programma simula un algoritmo di
routing Deficit Round Robin.");
        System.out.println("Vi verrà chiesto quante code generare e
la lunghezza del quanto di tempo.");
        System.out.println("Il numero dei pacchetti per ogni coda
verrà poi generato casualmente (compreso tra 1 e 10) ");
        System.out.println("così come il tempo a loro necessario
(compreso tra 1 e 1000).");
        System.out.println("Per inserire un numero di pacchetti fissi
per ogni coda, è sufficiente ");
        System.out.println("decommentare la parte relativa nel codice
sorgente.");
        System.out.println();

        // Quante code genero?
        Scanner input = new Scanner(System.in);
        System.out.println("Quante code devo generare?");
        while (!input.hasNextInt())
        {
            System.out.printf("Hai inserito un input non valido.
Ricordati di aggiungere solo numeri. Ritenta.\n");
            input.next();
        }
        int ncode = input.nextInt();

        /*
         * DECOMMENTARE SE SI VUOLE DECIDERE QUANTI PACCHETTI
GENERARE
         * Quanti pacchetti genero?
```

```
         * System.out.println("Quanti pacchetti devo generare per
ogni coda?");
         * while (!input.hasNextInt())
         * {
         * System.out.printf("Hai inserito un input non valido.
Ricordati di aggiungere solo numeri. Ritenta.\n");
         * input.next();
         * }
         * npacchetti = input.nextInt();
         */

        // Quanto sarà il quantum?
        System.out.println("Quanto è lungo il quanto di tempo?");
        while (!input.hasNextInt())
        {
            System.out.printf("Hai inserito un input non valido.
Ricordati di aggiungere solo numeri. Ritenta.\n");
            input.next();
        }
        quantum = input.nextInt();

        // Chiudiamo lo scanner
        input.close();

        // Genero code (che genereranno i pacchetti)
        Coda coda[] = new Coda[ncode];
        for (int i = 0; i < ncode; i++){
            coda[i] = new Coda(i+1);
            coda[i].start();
        }
    }
}
```

CODA.JAVA:

```
package progettoscm1213;

import java.util.ArrayList;
import java.util.Random;

/**
 *
 * @author Giovanni Dini
 */
public class Coda extends Thread {
    private int quantumattuale;
    private int contatorepacchetti = 1;
    private int id;
    private int deficit = 0;
    private int deficitiniziale = 0;

    public Coda(int id){
        this.id = id;
    }

    @Override
    public void run() {
        // Genero pacchetti
        ArrayList<Pacchetto> coda = new ArrayList<>();
        coda = generaPacchetti(coda, Main.npacchetti);
        // Controllo se coda vuota o meno
        // System.out.println("La coda contiene "+coda.size()+"
elementi.");
        // Settiamo il quantum
        quantumattuale = Main.quantum;
        System.out.println("--Coda "+this.id+" in partenza con
"+coda.size()+" pacchetti.");
        // Fin quando i pacchetti non sono finiti
        while (coda.isEmpty() == false){
            // Eseguo operazioni sulla coda
```

```
            service(coda);
        }

        // Termino
        System.out.println("Coda "+this.id+" - Pacchetti
terminati.");
    }

    public void service(ArrayList<Pacchetto> coda){
        // Prendo controllo
        Main.queue.lock();

        quantumattuale = Main.quantum + deficit;

        // Controllo se dimensioni pacchetto >= quanto di tempo
        if (coda.get(0).dimensione <= quantumattuale){

            // Prendo la dimensione del pacchetto
            int dimensione = coda.get(0).dimensione;

            // Calcolo il deficit
            deficitiniziale = deficit + Main.quantum;
            deficit = quantumattuale - dimensione;

            disegno(coda, deficit, deficitiniziale, this.id);
            //System.out.println("Deficit attuale: "+deficit);

            // Servo pacchetto
            System.out.println("CODA "+this.id+" - PACCHETTO
"+contatorepacchetti+" SERVITO. (D="+dimensione+" -
Q="+quantumattuale+")");

            // Rimuovo il pacchetto appena passato
            coda.remove(0);

            // Passo a elemento successivo della lista
            contatorepacchetti++;
```

```
            //System.out.println("La coda contiene ancora
"+coda.size()+" elementi.");
            // Cedo controllo
            Main.queue.unlock();
        } else {
            // Prendo la dimensione del pacchetto
            //int dimensione = coda.get(0).dimensione;

            // Aumento contatoreripetizioni per segnalare che avrò un
multiplo del quantum al prossimo giro
            //System.out.println("C"+this.id+"P"+contatorepacchetti+"
NON servito. (D="+dimensione+" - Q="+quantumattuale+")");

            // Calcolo il deficit
            deficitiniziale = deficit;
            deficit = quantumattuale;
            disegno(coda, deficit, deficitiniziale, this.id);

            // Cedo controllo
            Main.queue.unlock();
        }
    }

    public ArrayList<Pacchetto> generaPacchetti(ArrayList<Pacchetto>
coda, int npacchetti) {

        /*
         * DECOMMENTARE SE SI VUOLE DECIDERE QUANTI PACCHETTI
GENERARE (anche nella main)
         * (E COMMENTARE LE RIGHE SOTTO)
         * Genero pacchetti
         * Pacchetto p[] = new Pacchetto[npacchetti];
         */

        // Genero pacchetti random
        Random randomgen = new Random();
        npacchetti = randomgen.nextInt(10);
        Pacchetto p[] = new Pacchetto[npacchetti];
```

```
        for (int i = 0; i < npacchetti; i++){
            int dimensione = generaDimensione();
            p[i] = new Pacchetto(dimensione);
            //System.out.println(dimensione);
            // Inserisco pacchetti in coda
            coda.add(p[i]);
        }

        // Ritorno coda
        return coda;
    }

    public int generaDimensione() {
        // Generiamo una dimensione random per i pacchetti
        Random randomgen = new Random();
        int dimension = randomgen.nextInt(1000);
        return dimension;
    }

    public void disegno(ArrayList<Pacchetto> coda, int deficit, int
deficitiniziale, int id) {
        int dimensione0 = coda.get(0).dimensione;
        System.out.println("CODA "+id+"   |----"+dimensione0+"----|
->   DI: "+deficitiniziale+"   DF: "+deficit);
    }
}
```

PACCHETTO.JAVA:

```
package progettoscm1213;

/**
 *
 * @author Giovanni Dini
 */
public class Pacchetto {
    public int dimensione;
```

```
    // Costruttore del pacchetto
    Pacchetto(int dimensione) {
        this.dimensione = dimensione;
    }

}
```

Test

Nota: nel listato DI = Deficit Iniziale (all'inizio del turno), DF = Deficit Finale (alla fine del turno) e quando un pacchetto viene servito D = Dimensione del pacchetto e Q = Quanto a disposizione in quel turno.

```
/*****************************************/
* PROGRAMMA DI SIMULAZIONE DI UN ALGORITMO *
*           DEFICIT ROUND ROBIN            *
/*****************************************/
*    PROGETTO DI SISTEMI DI COMUNICAZIONE  *
*       MULTIMEDIALI - A.A. 2012/2013      *
/*****************************************/
*    GIOVANNI DINI - MATRICOLA 232274      *
/*****************************************/


Il programma simula un algoritmo di routing Deficit Round Robin.
Vi verrà chiesto quante code generare e la lunghezza del quanto di
tempo.
Il numero dei pacchetti per ogni coda verrà poi generato casualmente
(compreso tra 1 e 10)
così come il tempo a loro necessario (compreso tra 1 e 1000).
Per inserire un numero di pacchetti fissi per ogni coda, è
sufficiente
decommentare la parte relativa nel codice sorgente.


Quante code devo generare?
1
Quanto è lungo il quanto di tempo?
200
--Coda 1 in partenza con 6 pacchetti.
CODA 1   |----339----|   ->   DI: 0   DF: 200
CODA 1   |----339----|   ->   DI: 400   DF: 61
CODA 1 - PACCHETTO 1 SERVITO. (D=339 - Q=400)
CODA 1   |----755----|   ->   DI: 61   DF: 261
```

```
CODA 1   |----755----|   ->   DI: 261   DF: 461
CODA 1   |----755----|   ->   DI: 461   DF: 661
CODA 1   |----755----|   ->   DI: 861   DF: 106
CODA 1 - PACCHETTO 2 SERVITO. (D=755 - Q=861)
CODA 1   |----758----|   ->   DI: 106   DF: 306
CODA 1   |----758----|   ->   DI: 306   DF: 506
CODA 1   |----758----|   ->   DI: 506   DF: 706
CODA 1   |----758----|   ->   DI: 906   DF: 148
CODA 1 - PACCHETTO 3 SERVITO. (D=758 - Q=906)
CODA 1   |----941----|   ->   DI: 148   DF: 348
CODA 1   |----941----|   ->   DI: 348   DF: 548
CODA 1   |----941----|   ->   DI: 548   DF: 748
CODA 1   |----941----|   ->   DI: 948   DF: 7
CODA 1 - PACCHETTO 4 SERVITO. (D=941 - Q=948)
CODA 1   |----373----|   ->   DI: 7   DF: 207
CODA 1   |----373----|   ->   DI: 407   DF: 34
CODA 1 - PACCHETTO 5 SERVITO. (D=373 - Q=407)
CODA 1   |----121----|   ->   DI: 234   DF: 113
CODA 1 - PACCHETTO 6 SERVITO. (D=121 - Q=234)
Coda 1 - Pacchetti terminati.
```

Questo primo test mostra il funzionamento del procedimento di elaborazione dei pacchetti (per illustrarlo è stata generata una coda singola). L'algoritmo si comporta come previsto nel calcolo del deficit e del quanto per i turni successivi.

```
Quante code devo generare?
2
Quanto è lungo il quanto di tempo?
400
--Coda 1 in partenza con 6 pacchetti.
--Coda 2 in partenza con 8 pacchetti.
CODA 1   |----629----|   ->   DI: 0   DF: 400
CODA 2   |----793----|   ->   DI: 0   DF: 400
CODA 1   |----629----|   ->   DI: 800   DF: 171
CODA 1 - PACCHETTO 1 SERVITO. (D=629 - Q=800)
CODA 2   |----793----|   ->   DI: 800   DF: 7
CODA 2 - PACCHETTO 1 SERVITO. (D=793 - Q=800)
CODA 1   |----616----|   ->   DI: 171   DF: 571
CODA 2   |----437----|   ->   DI: 7   DF: 407
CODA 1   |----616----|   ->   DI: 971   DF: 355
CODA 1 - PACCHETTO 2 SERVITO. (D=616 - Q=971)
CODA 2   |----437----|   ->   DI: 807   DF: 370
CODA 2 - PACCHETTO 2 SERVITO. (D=437 - Q=807)
CODA 1   |----322----|   ->   DI: 755   DF: 433
CODA 1 - PACCHETTO 3 SERVITO. (D=322 - Q=755)
CODA 2   |----909----|   ->   DI: 370   DF: 770
CODA 1   |----94----|   ->   DI: 833   DF: 739
CODA 1 - PACCHETTO 4 SERVITO. (D=94 - Q=833)
CODA 2   |----909----|   ->   DI: 1170   DF: 261
CODA 2 - PACCHETTO 3 SERVITO. (D=909 - Q=1170)
CODA 1   |----453----|   ->   DI: 1139   DF: 686
CODA 1 - PACCHETTO 5 SERVITO. (D=453 - Q=1139)
CODA 2   |----886----|   ->   DI: 261   DF: 661
CODA 1   |----477----|   ->   DI: 1086   DF: 609
CODA 1 - PACCHETTO 6 SERVITO. (D=477 - Q=1086)
Coda 1 - Pacchetti terminati.
CODA 2   |----886----|   ->   DI: 1061   DF: 175
CODA 2 - PACCHETTO 4 SERVITO. (D=886 - Q=1061)
CODA 2   |----285----|   ->   DI: 575   DF: 290
CODA 2 - PACCHETTO 5 SERVITO. (D=285 - Q=575)
CODA 2   |----502----|   ->   DI: 690   DF: 188
CODA 2 - PACCHETTO 6 SERVITO. (D=502 - Q=690)
```

```
CODA 2   |----990----|   ->   DI: 188   DF: 588
CODA 2   |----990----|   ->   DI: 588   DF: 988
CODA 2   |----990----|   ->   DI: 1388   DF: 398
CODA 2 - PACCHETTO 7 SERVITO. (D=990 - Q=1388)
CODA 2   |----784----|   ->   DI: 798   DF: 14
CODA 2 - PACCHETTO 8 SERVITO. (D=784 - Q=798)
Coda 2 - Pacchetti terminati.
```

Questo semplice test si comporta come stabilito e mostra la corretta alternanza delle due code, oltre a confermare il test precedente.

```
Quante code devo generare?
4
Quanto è lungo il quanto di tempo?
120
--Coda 4 in partenza con 7 pacchetti.
--Coda 3 in partenza con 2 pacchetti.
CODA 4   |----20----|   ->   DI: 120   DF: 100
CODA 4 - PACCHETTO 1 SERVITO. (D=20 - Q=120)
--Coda 2 in partenza con 2 pacchetti.
--Coda 1 in partenza con 3 pacchetti.
CODA 3   |----927----|   ->   DI: 0   DF: 120
CODA 2   |----769----|   ->   DI: 0   DF: 120
CODA 1   |----519----|   ->   DI: 0   DF: 120
CODA 4   |----175----|   ->   DI: 220   DF: 45
CODA 4 - PACCHETTO 2 SERVITO. (D=175 - Q=220)
CODA 3   |----927----|   ->   DI: 120   DF: 240
CODA 2   |----769----|   ->   DI: 120   DF: 240
CODA 1   |----519----|   ->   DI: 120   DF: 240
CODA 4   |----458----|   ->   DI: 45   DF: 165
CODA 3   |----927----|   ->   DI: 240   DF: 360
CODA 2   |----769----|   ->   DI: 240   DF: 360
CODA 1   |----519----|   ->   DI: 240   DF: 360
CODA 4   |----458----|   ->   DI: 165   DF: 285
CODA 3   |----927----|   ->   DI: 360   DF: 480
CODA 2   |----769----|   ->   DI: 360   DF: 480
CODA 1   |----519----|   ->   DI: 360   DF: 480
CODA 4   |----458----|   ->   DI: 285   DF: 405
CODA 3   |----927----|   ->   DI: 480   DF: 600
CODA 2   |----769----|   ->   DI: 480   DF: 600
CODA 1   |----519----|   ->   DI: 600   DF: 81
CODA 1 - PACCHETTO 1 SERVITO. (D=519 - Q=600)
CODA 4   |----458----|   ->   DI: 525   DF: 67
CODA 4 - PACCHETTO 3 SERVITO. (D=458 - Q=525)
CODA 3   |----927----|   ->   DI: 600   DF: 720
CODA 2   |----769----|   ->   DI: 600   DF: 720
CODA 1   |----193----|   ->   DI: 201   DF: 8
CODA 1 - PACCHETTO 2 SERVITO. (D=193 - Q=201)
```

```
CODA 4   |----727----|   ->   DI: 67   DF: 187
CODA 3   |----927----|   ->   DI: 720   DF: 840
CODA 2   |----769----|   ->   DI: 840   DF: 71
CODA 2 - PACCHETTO 1 SERVITO. (D=769 - Q=840)
CODA 1   |----986----|   ->   DI: 8   DF: 128
CODA 4   |----727----|   ->   DI: 187   DF: 307
CODA 3   |----927----|   ->   DI: 960   DF: 33
CODA 3 - PACCHETTO 1 SERVITO. (D=927 - Q=960)
CODA 2   |----916----|   ->   DI: 71   DF: 191
CODA 1   |----986----|   ->   DI: 128   DF: 248
CODA 4   |----727----|   ->   DI: 307   DF: 427
CODA 3   |----707----|   ->   DI: 33   DF: 153
CODA 2   |----916----|   ->   DI: 191   DF: 311
CODA 1   |----986----|   ->   DI: 248   DF: 368
CODA 4   |----727----|   ->   DI: 427   DF: 547
CODA 3   |----707----|   ->   DI: 153   DF: 273
CODA 2   |----916----|   ->   DI: 311   DF: 431
CODA 1   |----986----|   ->   DI: 368   DF: 488
CODA 4   |----727----|   ->   DI: 547   DF: 667
CODA 3   |----707----|   ->   DI: 273   DF: 393
CODA 2   |----916----|   ->   DI: 431   DF: 551
CODA 1   |----986----|   ->   DI: 488   DF: 608
CODA 4   |----727----|   ->   DI: 787   DF: 60
CODA 4 - PACCHETTO 4 SERVITO. (D=727 - Q=787)
CODA 3   |----707----|   ->   DI: 393   DF: 513
CODA 2   |----916----|   ->   DI: 551   DF: 671
CODA 1   |----986----|   ->   DI: 608   DF: 728
CODA 4   |----383----|   ->   DI: 60   DF: 180
CODA 3   |----707----|   ->   DI: 513   DF: 633
CODA 2   |----916----|   ->   DI: 671   DF: 791
CODA 1   |----986----|   ->   DI: 728   DF: 848
CODA 4   |----383----|   ->   DI: 180   DF: 300
CODA 3   |----707----|   ->   DI: 753   DF: 46
CODA 3 - PACCHETTO 2 SERVITO. (D=707 - Q=753)
Coda 3 - Pacchetti terminati.
CODA 2   |----916----|   ->   DI: 791   DF: 911
CODA 1   |----986----|   ->   DI: 848   DF: 968
```

```
CODA 4   |----383----|   ->   DI: 420   DF: 37
CODA 4 - PACCHETTO 5 SERVITO. (D=383 - Q=420)
CODA 2   |----916----|   ->   DI: 1031   DF: 115
CODA 2 - PACCHETTO 2 SERVITO. (D=916 - Q=1031)
Coda 2 - Pacchetti terminati.
CODA 1   |----986----|   ->   DI: 1088   DF: 102
CODA 1 - PACCHETTO 3 SERVITO. (D=986 - Q=1088)
Coda 1 - Pacchetti terminati.
CODA 4   |----459----|   ->   DI: 37   DF: 157
CODA 4   |----459----|   ->   DI: 157   DF: 277
CODA 4   |----459----|   ->   DI: 277   DF: 397
CODA 4   |----459----|   ->   DI: 517   DF: 58
CODA 4 - PACCHETTO 6 SERVITO. (D=459 - Q=517)
CODA 4   |----108----|   ->   DI: 178   DF: 70
CODA 4 - PACCHETTO 7 SERVITO. (D=108 - Q=178)
Coda 4 - Pacchetti terminati.
```

Questo test con più code conferma il buon funzionamento.

Quante code devo generare?

ciao**--

Hai inserito un input non valido. Ricordati di aggiungere solo numeri. Ritenta.

234ç°

Hai inserito un input non valido. Ricordati di aggiungere solo numeri. Ritenta.

1

Quanto è lungo il quanto di tempo?

-.,avc

Hai inserito un input non valido. Ricordati di aggiungere solo numeri. Ritenta.

200

--Coda 1 in partenza con 9 pacchetti.

CODA 1 |----538----| -> DI: 0 DF: 200

CODA 1 |----538----| -> DI: 200 DF: 400

CODA 1 |----538----| -> DI: 600 DF: 62

CODA 1 - PACCHETTO 1 SERVITO. (D=538 - Q=600)

CODA 1 |----938----| -> DI: 62 DF: 262

CODA 1 |----938----| -> DI: 262 DF: 462

CODA 1 |----938----| -> DI: 462 DF: 662

CODA 1 |----938----| -> DI: 662 DF: 862

CODA 1 |----938----| -> DI: 1062 DF: 124

CODA 1 - PACCHETTO 2 SERVITO. (D=938 - Q=1062)

CODA 1 |----163----| -> DI: 324 DF: 161

CODA 1 - PACCHETTO 3 SERVITO. (D=163 - Q=324)

CODA 1 |----387----| -> DI: 161 DF: 361

CODA 1 |----387----| -> DI: 561 DF: 174

CODA 1 - PACCHETTO 4 SERVITO. (D=387 - Q=561)

CODA 1 |----903----| -> DI: 174 DF: 374

CODA 1 |----903----| -> DI: 374 DF: 574

CODA 1 |----903----| -> DI: 574 DF: 774

CODA 1 |----903----| -> DI: 974 DF: 71

CODA 1 - PACCHETTO 5 SERVITO. (D=903 - Q=974)

CODA 1 |----509----| -> DI: 71 DF: 271

CODA 1 |----509----| -> DI: 271 DF: 471

CODA 1 |----509----| -> DI: 671 DF: 162

CODA 1 - PACCHETTO 6 SERVITO. (D=509 - Q=671)

```
CODA 1   |----664----|   ->   DI: 162   DF: 362
CODA 1   |----664----|   ->   DI: 362   DF: 562
CODA 1   |----664----|   ->   DI: 762   DF: 98
CODA 1 - PACCHETTO 7 SERVITO. (D=664 - Q=762)
CODA 1   |----505----|   ->   DI: 98   DF: 298
CODA 1   |----505----|   ->   DI: 298   DF: 498
CODA 1   |----505----|   ->   DI: 698   DF: 193
CODA 1 - PACCHETTO 8 SERVITO. (D=505 - Q=698)
CODA 1   |----402----|   ->   DI: 193   DF: 393
CODA 1   |----402----|   ->   DI: 593   DF: 191
CODA 1 - PACCHETTO 9 SERVITO. (D=402 - Q=593)
Coda 1 - Pacchetti terminati.
```

Questo test mostra il buon funzionamento della validazione degli input (essendo solo questo lo scopo, è stata generata una sola coda).

Note finali

PIATTAFORMA DI SVILUPPO:

Il software è stato sviluppato con:

```
NetBeans IDE 7.2 (Build 201207301726)
```

Versione di Java in uso:

```
1.7.0_07; Java HotSpot(TM) 64-Bit Server VM 23.3-b01
```

Sistema operativo:

```
Versione 10.7.5 di Mac OS X in esecuzione su x86_64; US-ASCII; it_IT
(nb)
```

LICENZA E DISPONIBILITÀ DEL CODICE:

Il codice è in licenza Creative Commons 3.0 (CC-BY-NC-SA 3.0), interamente disponibile con tutti i vari commit a questo indirizzo:

https://github.com/gioggi2002/ProgettoSCM1213