Commenti

Il vero osso duro è rappresentato dall'inattività degli studenti, sia in aula che a casa: seguono poco e non fanno esercizi.

Inizio a pensare che sia convinti che per il solo fatto di essere seduti in aula saranno magicamente indottrinati.

Un esempio riassuntivo

Esempio: Tombola!

Tombola!

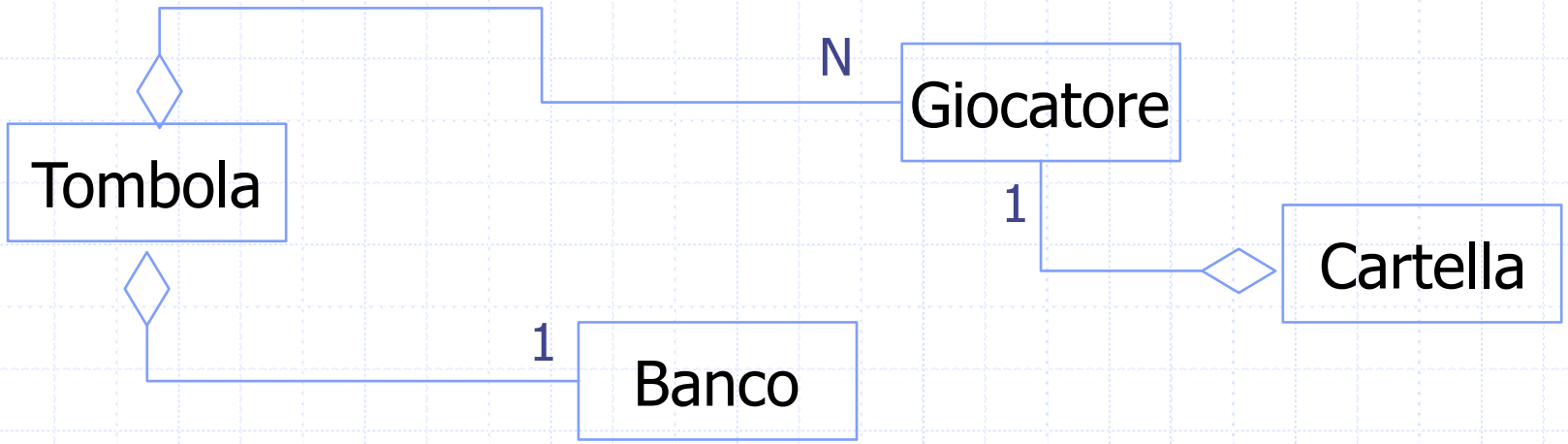
	17	22		45		66		83
1	19		30	48			70	
		29	37		53		74	86

Tombola - esercizio

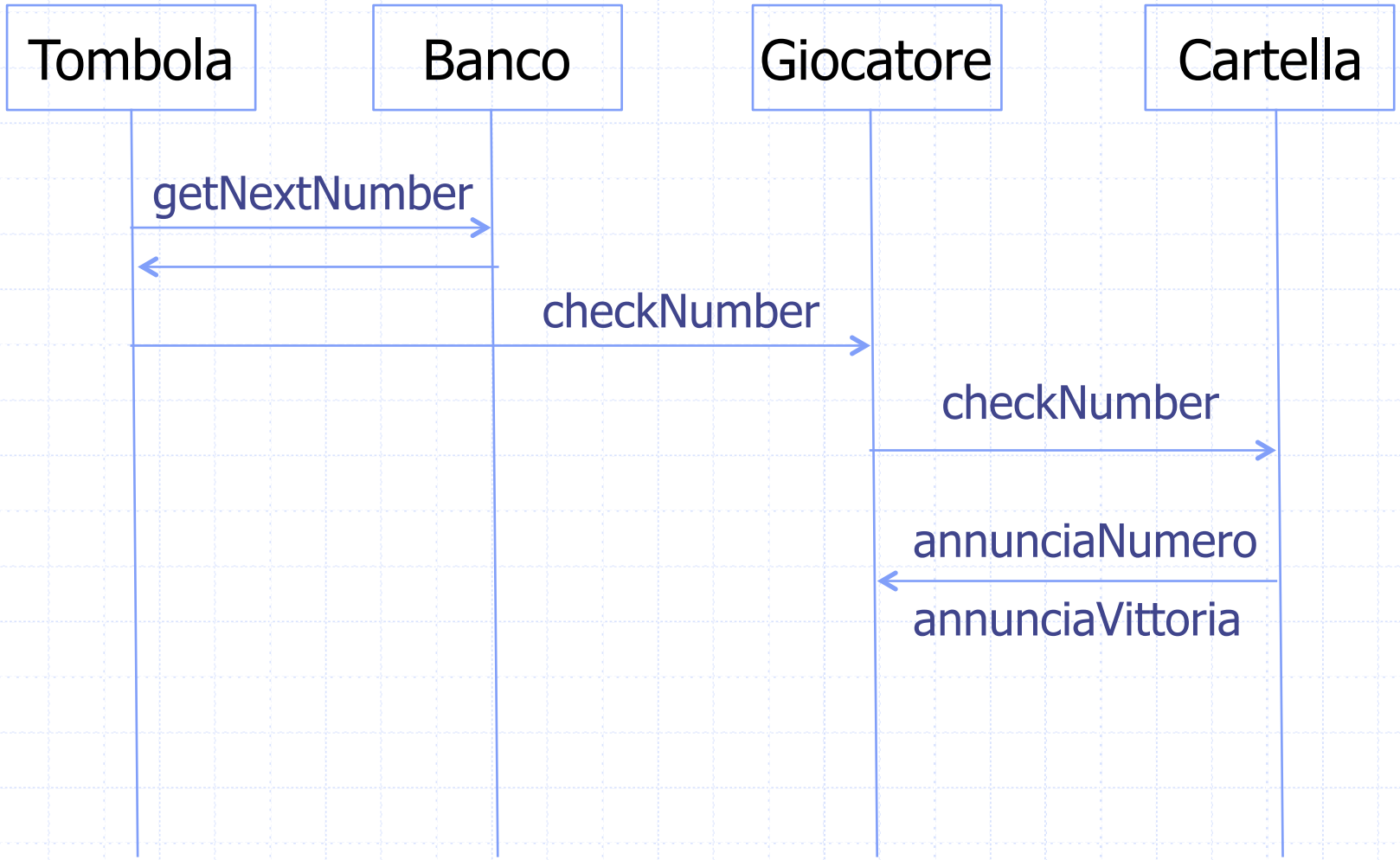
Esercizio:

Modificare il codice aggiungendo un numero arbitario di giocatori, ciascuno con un numero arbitario di cartelle

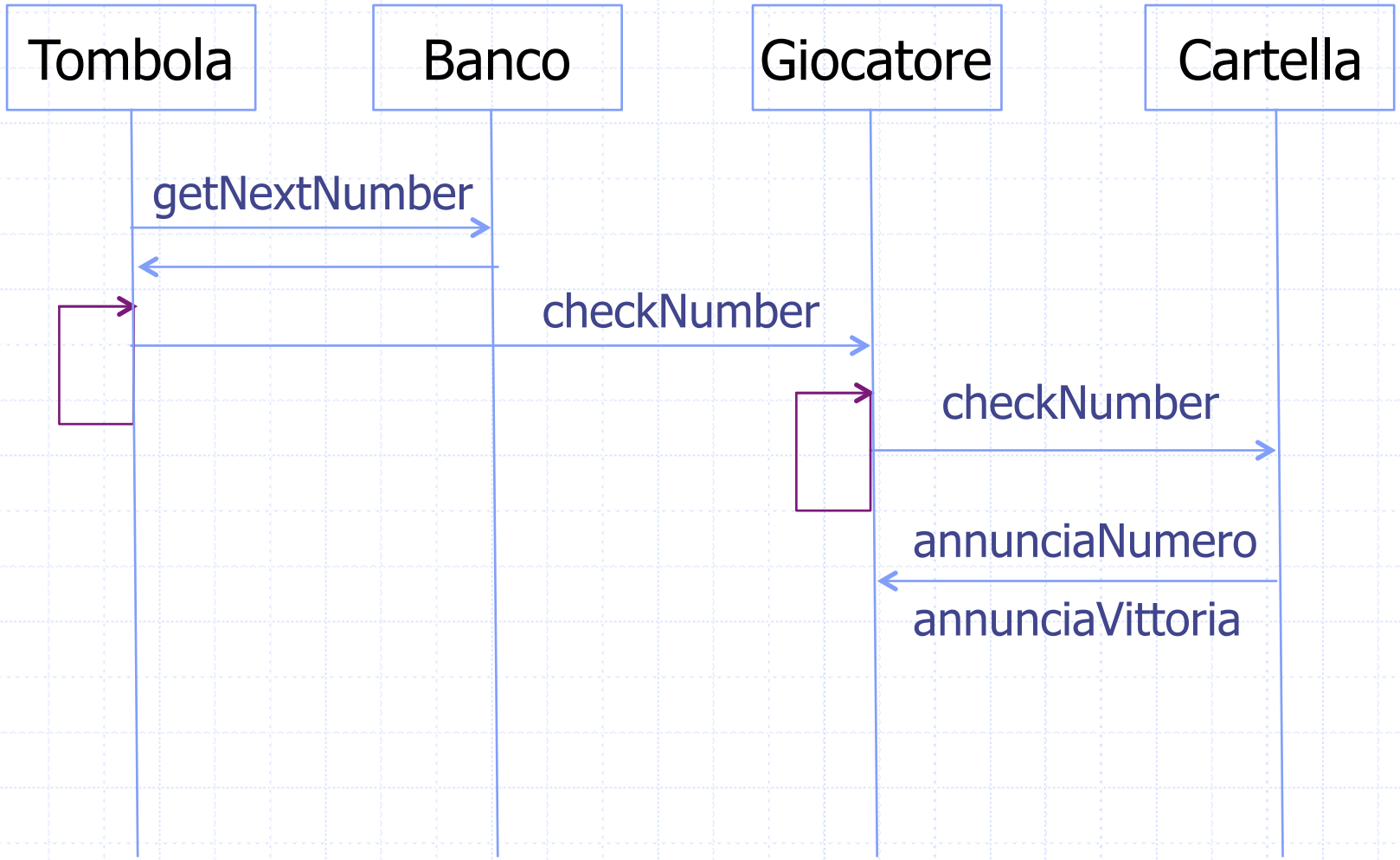
Class diagram



Activity diagram (swim lanes)



Activity diagram (swim lanes)



Common

```
package tombola;
import java.util.Random;
public class Common {
    static final int NCELLS=3;
    static final int MAXNUM=10;
    static final Random generatore =
        new  Random(System.currentTimeMillis());
}
```

Banco

```
package tombola;

import java.util.LinkedList;
import java.util.List;

public class Banco {
    List sacchetto;

    public Banco() {
        sacchetto= new LinkedList();
        for (int i=1; i<=Common.MAXNUM;i++) {
            sacchetto.add(new Integer(i));
        }
    }
```

Banco

```
public int getNextNumber() {
        if (sacchetto.size()==0) {
            System.out.println("NUMERI FINITI!");
            System.exit(1);
        }
        int index=Common.generatore.nextInt(sacchetto.size());
        Integer num=(Integer)sacchetto.get(index);
        sacchetto.remove(index);
        System.out.println("=====> ESTRATTO: "+num );
        return num.intValue();
    }
```

Banco – unit test

```
    public static void main(String[] args) {
        Banco banco = new Banco();
        while (true) {
            banco.getNextNumber();
        }
    }
}
```

```
run:
====> ESTRATTO: 8
====> ESTRATTO: 2
====> ESTRATTO: 6
====> ESTRATTO: 3
====> ESTRATTO: 7
====> ESTRATTO: 9
====> ESTRATTO: 4
====> ESTRATTO: 5
====> ESTRATTO: 10
====> ESTRATTO: 1
NUMERI FINITI!
Java Result: 1
```

Cartella

```
package tombola;

import java.util.HashSet;
import java.util.Iterator;

public class Cartella {
    private HashSet numeri = new HashSet();
    private HashSet mancanti = new HashSet();
    private Giocatore proprietario=null;
    private int id=0;
    static int nCartelle=0;

    Cartella(Giocatore g) {
        id=++nCartelle;
        proprietario=g;
        for (int i = 1; i <= Common.NCELLS; i++) {
            boolean creatoNuovoNumero = false;
            do {
                int x = Common.generatore.nextInt(Common.MAXNUM)+1;
                creatoNuovoNumero = numeri.add(new Integer(x));
                if (creatoNuovoNumero) System.out.println("aggiunto "+ x);
            } while (!creatoNuovoNumero);
        }
        mancanti.addAll(numeri);
    }
```

Cartella

```
public boolean checkNumber(int x) {
    boolean result = mancanti.remove(new Integer(x));
    if(proprietario!=null) {
        if (result) proprietario.annunciaNumero(x, id);
        if (mancanti.isEmpty()) proprietario.annunciaVittoria(id);
    }
    return result;
}

private void print(HashSet list) {
    Iterator iter = list.iterator();
    while (iter.hasNext()) {
        System.out.print(iter.next()+" ");
    }
    System.out.println();
}
public void printOriginale() {print(numeri);}
public void printCurrent() {print(mancanti);}
```

Cartella – unit test

```
public static void main(String a[]) {
    Cartella x=new Cartella(null);
    x.printCurrent();
    while (!x.mancanti.isEmpty()) {
        int k=Common.generatore.nextInt(Common.MAXNUM)+1;
        if (x.checkNumber(k)) System.out.println("==> Trovato "+k);
        else System.out.println(k);
    }
    System.out.println("Finito!");
    x.printOriginale();
  }
}
```

```
aggiunto 2
aggiunto 7
aggiunto 9
2 7 9
==> Trovato 7
```

```
4
...
6
==> Trovato 9
7
```

```
...
1
==> Trovato 2
Finito!
2 7 9
```

Player

```
package tombola;
public class Giocatore {
    public String name;
    private Cartella cartella;
    Giocatore(String name){
        this.name=name;
        cartella=new Cartella(this);
    }
    void checkNumber(int x){
        cartella.checkNumber(x);
    }
```

Player

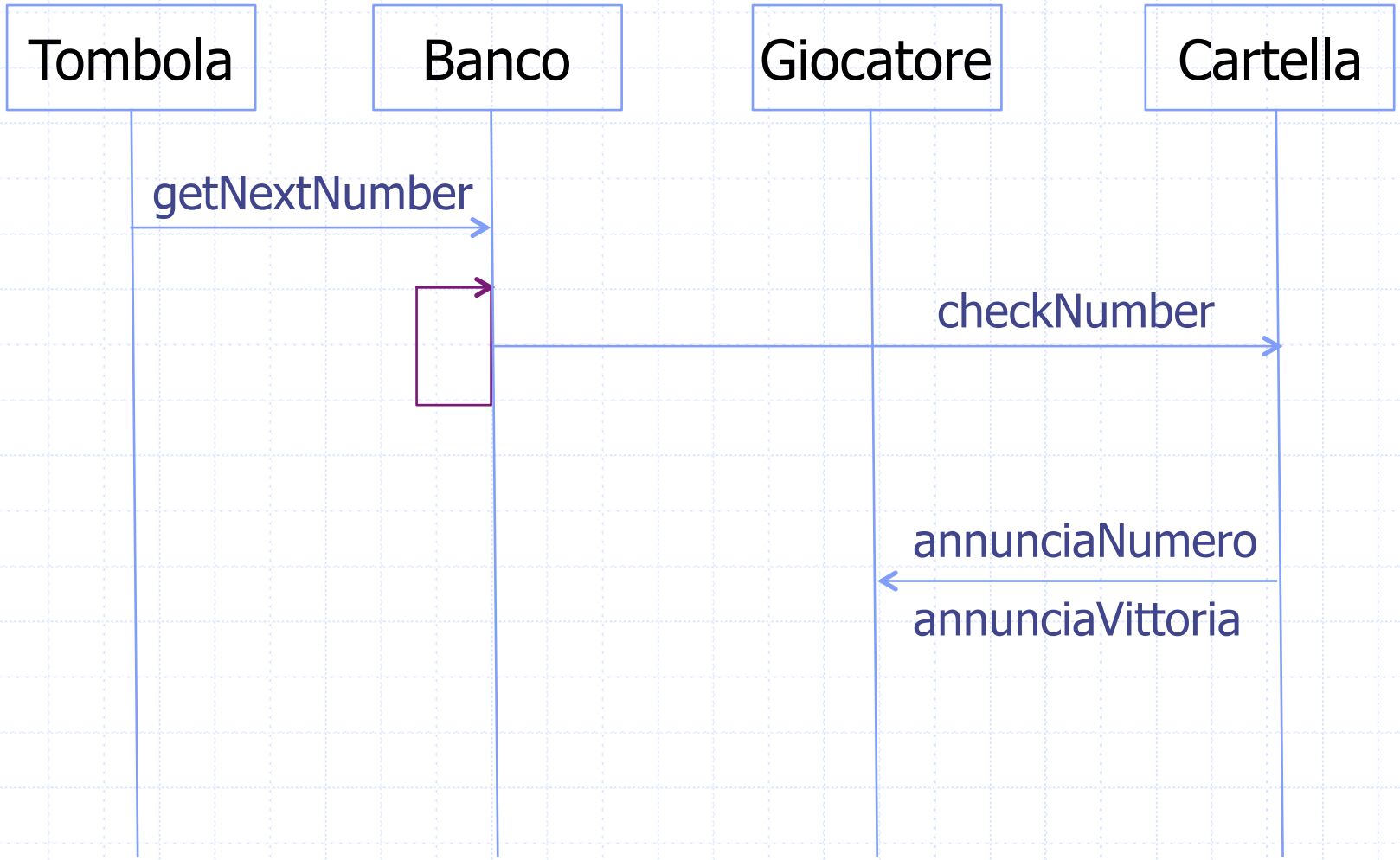
```
 void annunciaNumero(int num, int cartellaId){
     System.out.println(name+" ha il numero
"+num+" in cartella "+cartellaId);
 }
 void annunciaVittoria(int cartellaId) {
     System.out.println(name+" ha vinto con
cartella "+cartellaId);
     cartella.printOriginale();
     System.exit(1);
 }
}
```

Tombola

```
package tombola;

public class Tombola {
    public Tombola() {
        Banco banco = new Banco();

        Giocatore p = new Giocatore("Pippo");
        while (true) {
            int x = banco.getNextNumber();
            System.out.println(
                "Il numero estratto é " + x);
            p.checkNumber(x);
        }
    }

  public static void main(String[] args) {
        Tombola x=new Tombola();
    }
}
```

```
aggiunto 5
aggiunto 4
aggiunto 1
====> ESTRATTO: 1
Il numero estratto é 1
Pippo ha il numero 1 in cartella 1
====> ESTRATTO: 5
Il numero estratto é 5
Pippo ha il numero 5 in cartella 1
====> ESTRATTO: 7
Il numero estratto é 7
====> ESTRATTO: 10
Il numero estratto é 10
====> ESTRATTO: 2
Il numero estratto é 2
====> ESTRATTO: 4
Il numero estratto é 4
Pippo ha il numero 4 in cartella 1
Pippo ha vinto con cartella 1
1 4 5
```

Activity diagram (swim lanes)

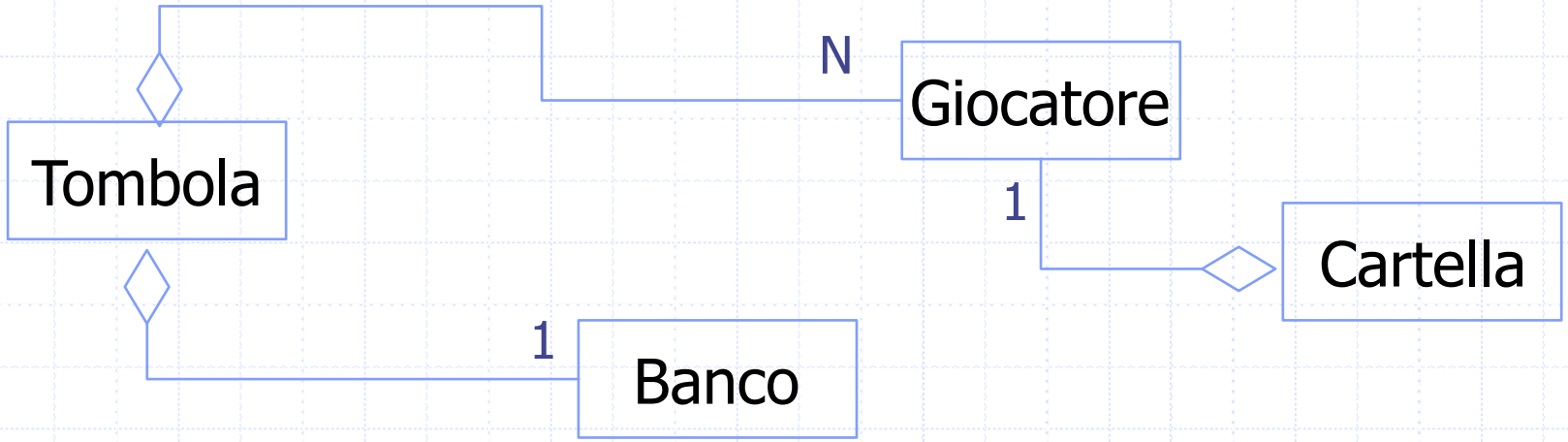


Listener

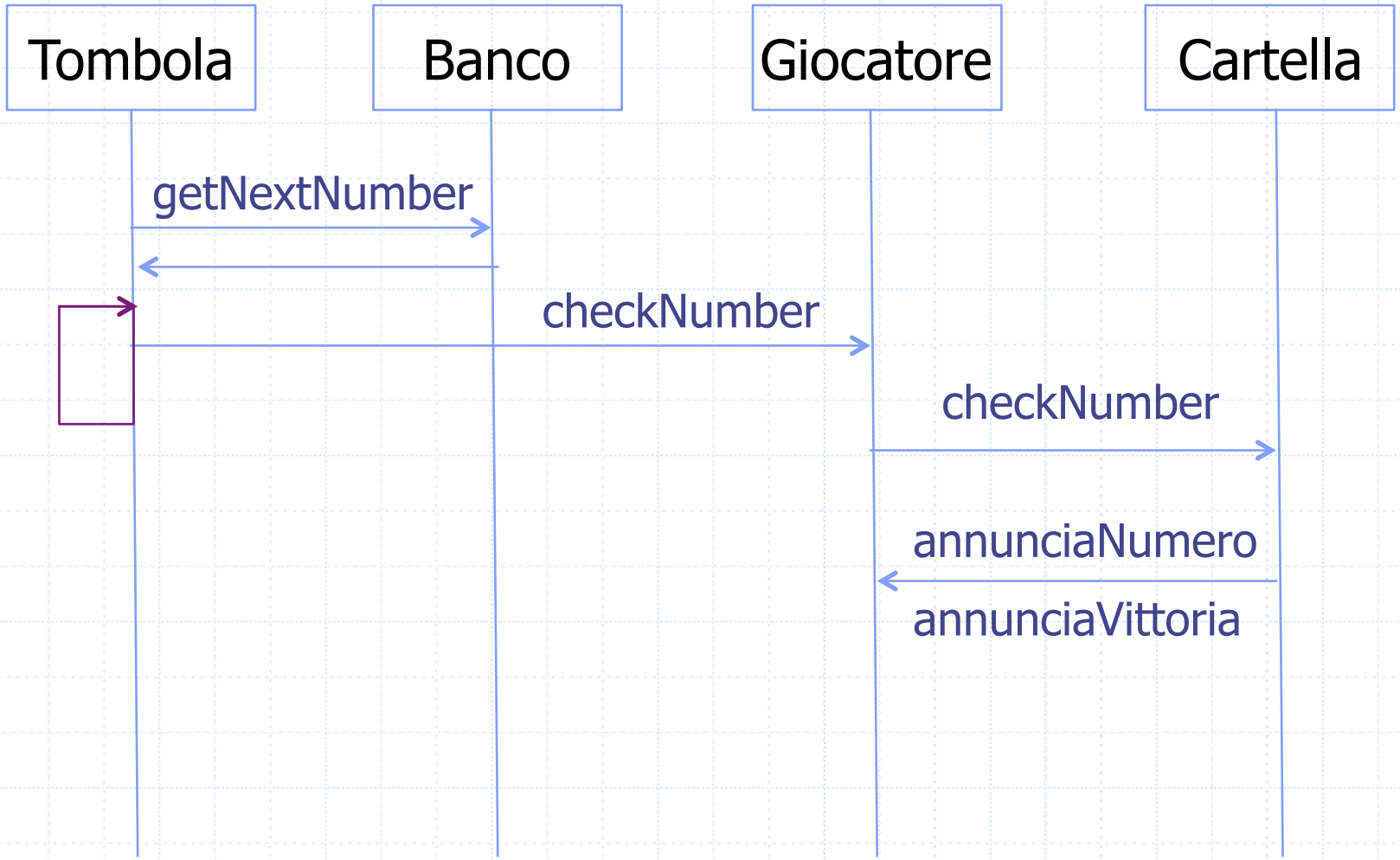
Introduciamo la nozione di "ascoltatore"

secondo il paradigma "publish & subscribe"

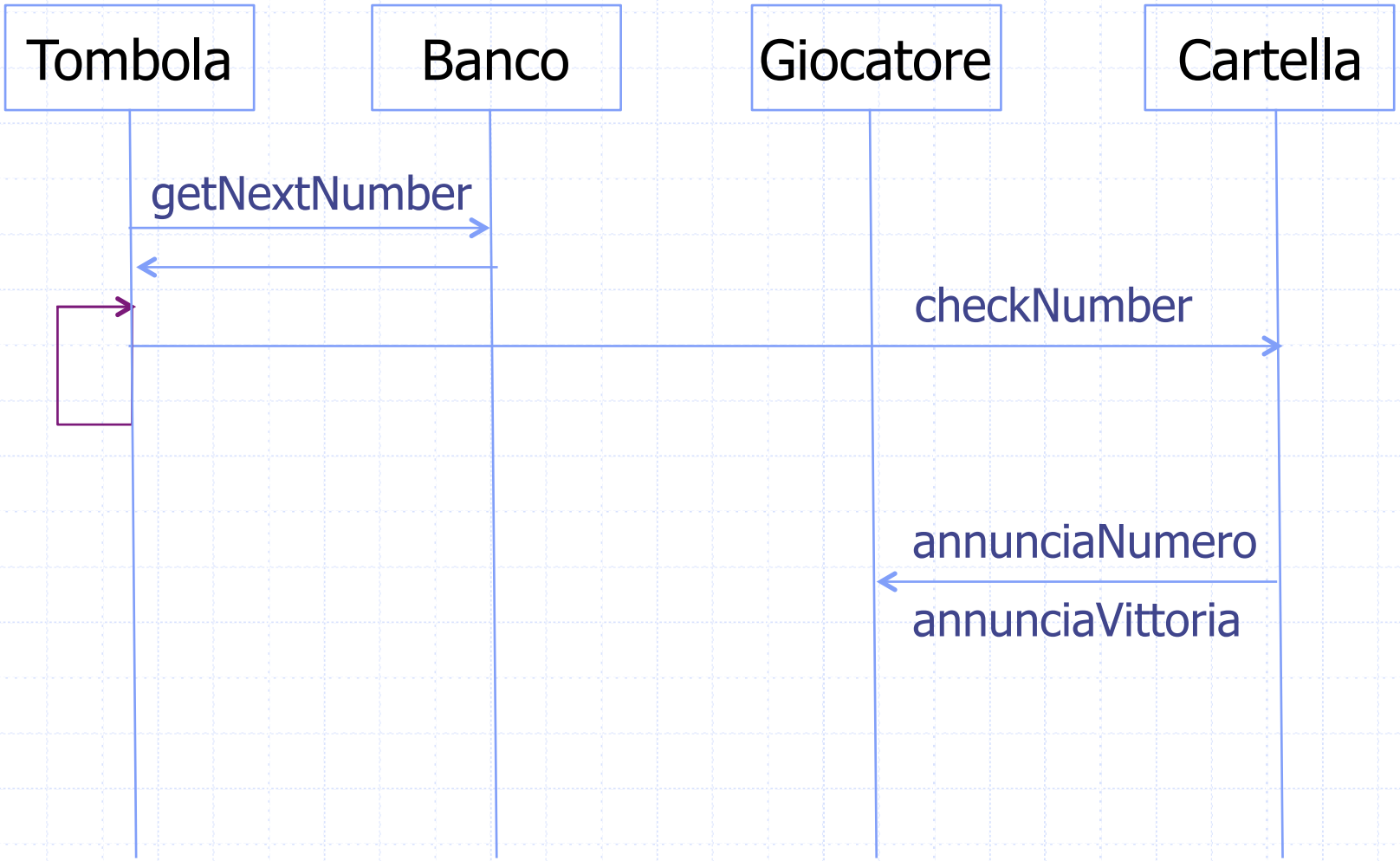
Class diagram



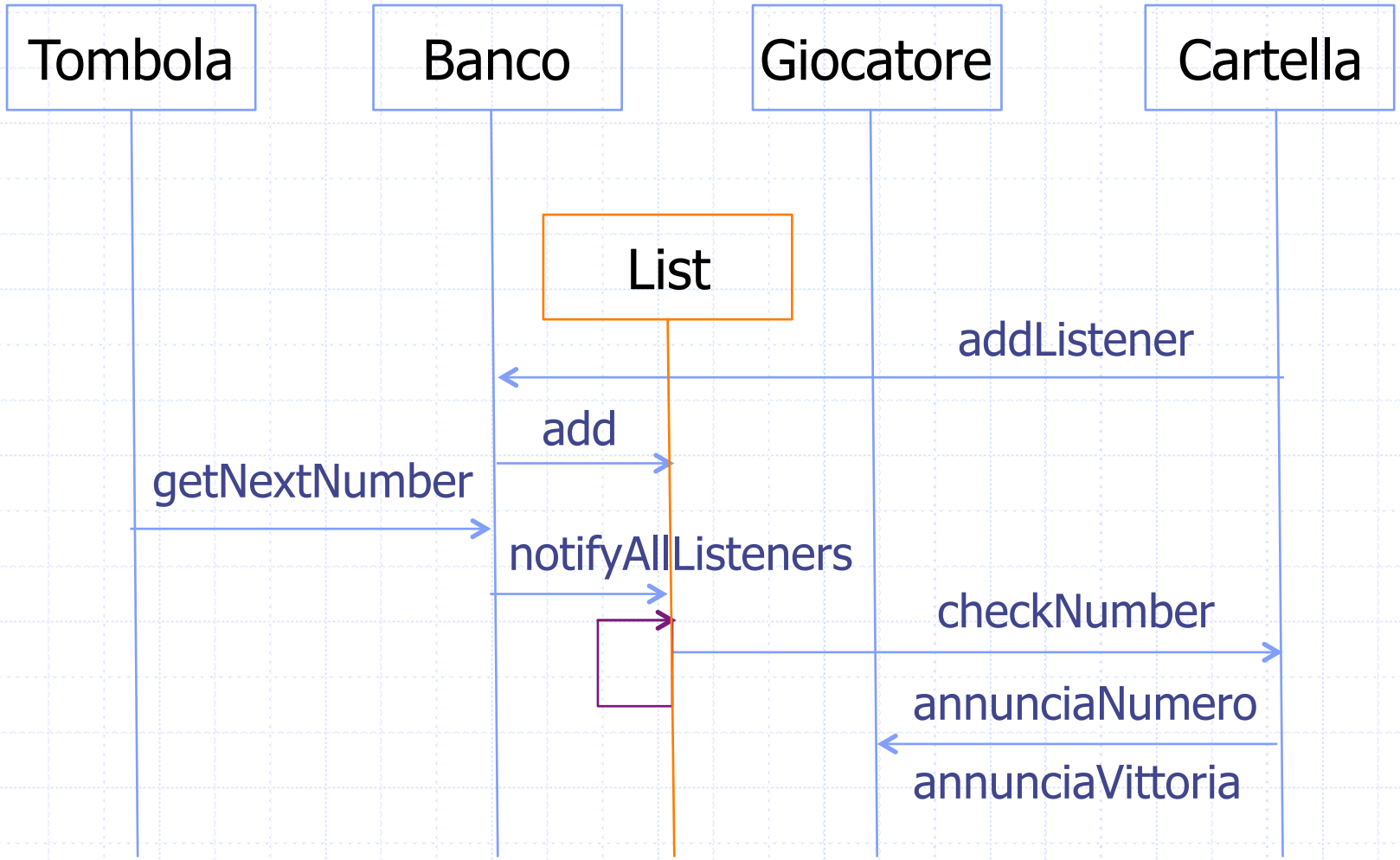
Activity diagram (swim lanes)



Activity diagram (swim lanes)



Activity diagram (swim lanes)



Common


```
package tombola;
import java.util.Random;
public class Common {
    static final int NCELLS=3;
    static final int MAXNUM=10;
    static final Random generatore =
        new  Random(System.currentTimeMillis());
}
```

Giocatore


```
package tombola;
public class Giocatore {
    public String name;
    private Cartella cartella;
    Giocatore(String name){
        this.name=name;
        cartella=new Cartella(this);
    }
    void checkNumber(int x){
        cartella.checkNumber(x);
    }
```

Giocatore

```
    void annunciaNumero(int num, int cartellaId){
        System.out.println(name+" ha il numero
    "+num+" in cartella "+cartellaId);
    }
    void annunciaVittoria(int cartellaId) {
        System.out.println(name+" ha vinto con
    cartella "+cartellaId);
        cartella.printOriginale();
        System.exit(1);
    }
}
```

Cartella

```
package tombola;

import java.util.HashSet;
import java.util.Iterator;

public class Cartella {
    private HashSet numeri = new HashSet();
    private HashSet mancanti = new HashSet();
    private Giocatore proprietario=null;
    private int id=0;
    static int nCartelle=0;

    Cartella(Giocatore g) {
        id=++nCartelle;
        proprietario=g;
        for (int i = 1; i <= Common.NCELLS; i++) {
            boolean creatoNuovoNumero = false;
            do {
                int x = Common.generatore.nextInt(Common.MAXNUM)+1;
                creatoNuovoNumero = numeri.add(new Integer(x));
                if (creatoNuovoNumero) System.out.println("aggiunto "+ x);
            } while (!creatoNuovoNumero);
        }
        mancanti.addAll(numeri);
    }
```

Cartella

```
public boolean checkNumber(int x) {
    boolean result = mancanti.remove(new Integer(x));
    if(proprietario!=null) {
        if (result) proprietario.annunciaNumero(x, id);
        if (mancanti.isEmpty()) proprietario.annunciaVittoria(id);
    }
    return result;
}

private void print(HashSet list) {
    Iterator iter = list.iterator();
    while (iter.hasNext()) {
        System.out.print(iter.next()+" ");
    }
    System.out.println();
}
public void printOriginale() {print(numeri);}
public void printCurrent() {print(mancanti);}
}
```

Banco

```
package tombola;

import java.util.LinkedList;
import java.util.List;
```

```
public class Banco {
    List sacchetto;
    List cartelle;

    public Banco() {
        cartelle= new LinkedList();
        sacchetto= new LinkedList();
        for (int i=1; i<=Common.MAXNUM;i++) {
            sacchetto.add(new Integer(i));
        }
    }
```

Banco – Lista di ascoltatori

```
void addListener(Cartella c){
        cartelle.add(c);
    }
void removeListener(Cartella c){
        cartelle.remove(c);
    }
private void notifyAllListeners(int x){
        Iterator iter=cartelle.iterator();
        while (iter.hasNext()) {
            ((Cartella) (iter.next())).checkNumber(x);
     }  }
```

Banco

```
public int getNextNumber() {
        if (sacchetto.size()==0) {
            System.out.println("NUMERI FINITI!");
            System.exit(1);
        }
        int index=Common.generatore.nextInt(sacchetto.size());
        Integer num=(Integer)sacchetto.get(index);
        sacchetto.remove(index);
        System.out.println("=====> ESTRATTO: "+num );
        notifyAllListeners(num.intValue());
        return num.intValue();
    }
}
```

Tombola

```
package tombola;

public class Tombola {
   public Tombola() {
        Banco banco = new Banco();

        Giocatore g1 = new Giocatore("Pippo");
        Cartella cartella=new Cartella(g1);
        banco.addListener(cartella);
        banco.addListener(new Cartella(g1));

        Giocatore g2 = new Giocatore("Pluto");
        banco.addListener(new Cartella(g2));
        while (true) {
            int x = banco.getNextNumber();
            System.out.println("Il numero estratto é " + x);
        }
    }

  public static void main(String[] args) {
        Tombola x=new Tombola();
    }
}
```

run!

```
Cartella 1 per giocatore Pippo : 5 3 6
Cartella 2 per giocatore Pippo : 8 7 10
Cartella 3 per giocatore Pluto : 1 5 4
====> ESTRATTO: 7
Pippo ha il numero 7 in cartella 2
Il numero estratto é 7
====> ESTRATTO: 5
Pippo ha il numero 5 in cartella 1
Pluto ha il numero 5 in cartella 3
Il numero estratto é 5
====> ESTRATTO: 9
Il numero estratto é 9
```

```
====> ESTRATTO: 6
Pippo ha il numero 6 in cartella 1
Il numero estratto é 6
====> ESTRATTO: 3
Pippo ha il numero 3 in cartella 1
Pippo ha vinto con cartella 1
3 5 6
```

Java

Generics

Uso di Generics nelle API di Java

```
// Removes 4-letter words from c. Elements must be strings
   static void expurgate(Collection c) {
      for (Iterator i = c.iterator(); i.hasNext(); )
        if (((String) i.next()).length() == 4)
          i.remove();
   }
```

Problemi?

Uso di Generics nelle API di Java

```
// Removes 4-letter words from c. Elements must be strings
   static void expurgate(Collection c) {
      for (Iterator i = c.iterator(); i.hasNext(); )
        if (((String) i.next()).length() == 4)
          i.remove();
   }
```

Here is the same example modified to use generics:

```
   // Removes the 4-letter words from c
   static void expurgate(Collection<String> c) {
      for (Iterator<String> i = c.iterator(); i.hasNext(); )
        if (i.next().length() == 4)
          i.remove();
   }
```

A partire da Java 5 molte classi sono state riscritte usando i generics

Vantaggi

No cast

Fail quick!

Definizione

A generic type is a reference type that has one or more type parameters. In the definition of the generic type, the type parameter section follows the type name. It is a comma separated list of identifiers and is delimited by angle brackets.

```
class Pair<X,Y>  {
  private X first;
  private Y second;   public Pair(X a1, Y a2) {
    first  = a1;
    second = a2;
  }
  public X getFirst()  { return first; }
  public Y getSecond() { return second; }
  public void setFirst(X arg)  { first = arg; }
  public void setSecond(Y arg) { second = arg; }
}
```

Definizione - continua

The class Pair has two type parameters X and Y .

They are replaced by type arguments when the generic type Pair is instantiated.

For instance, in the declaration Pair<String, Date> the type parameter X is replaced by the type argument String and Y is replaced by Date .

The scope of the identifiers X and Y is the entire definition of the class. In this scope the two type parameters X and Y are used like they were types (with some restrictions).

Esempio

```
public void printPair( Pair<String,Long> pair) {
        System.out.println("("+pair.getFirst()+","
            +pair.getSecond()+")");
  }

  Pair<String,Long> limit =
        new Pair<String,Long> ("maximum",1024L);
  printPair(limit);
```

Pbm: Generics are not usable...

- **for creation of arrays**
- **in an instanceof expression**

Generic Pila

```
public class Pila <T> {
  protected int size;
  protected int defaultGrowthSize=5;
  protected int marker;
  protected T contenuto[];
  protected final int initialSize=3;

  public Pila () {
     size=initialSize;
     marker=0;
     // contenuto=new T[size]); NO!
     contenuto=(T[])(new Object[size]);
  }
```

Generic Pila

```
private void cresci(int dim){
  T temp[ ]=(T[])(new Object[size]);
  for (int k=0;k<size;k++)
     temp[k]=contenuto[k];
  contenuto=(T[])(new Object[size+defaultGrowthSize]);
  for (int k=0;k<size;k++)
     contenuto[k]=temp[k];
  size+=defaultGrowthSize;
}
```

Generic Pila

```
public final void inserisci(T k) {
  if (marker == size) {
    cresci(defaultGrowthSize);
  }
  contenuto[marker] = k;
  marker++;
}

public T estrai() {
  assert(marker > 0):"Estrazione da Pila vuota";
  return contenuto[--marker];
}
```

Generic Pila

```
public static void main(String args[]) {
    int dim = 10;
    Pila<Integer> s = new Pila<Integer>();
    for (int k = 0; k < dim; k++) {
      s.inserisci(new Integer(k));
    }
    for (int k = 0; k < 3 * dim; k++) {
      Integer w = s.estrai();
     // Integer w = (Integer) s.estrai();
      System.out.println(w);
    }
  }

}
```

Note: Pila.java uses unchecked or unsafe operations.

Generic Pila

```
public static void main(String args[]) {
    int dim = 10;
    Pila<Integer> s = new Pila<Integer>();
    for (int k = 0; k < dim; k++) {
      s.inserisci(new String("pippo")));
    }
    for (int k = 0; k < 3 * dim; k++) {
      Integer w = s.estrai();
     // Integer w = (Integer) s.estrai();
      System.out.println(w);
    }
  }
}
```

```
Pila.java:43: inserisci(java.lang.Integer)
in Pila<java.lang.Integer> cannot be
applied to (java.lang.String)
    s.inserisci(new String("pippo"));
     ^
```

Wildcard instantiation

```
public void printPair( Pair<?,?> pair) {
        System.out.println("("+pair.getFirst()+","
            +pair.getSecond()+")");
  }

  Pair<?,?> limit =
        new Pair<String,Long> ("maximum",1024L);
  printPair(limit);
```

Referenze su generics:

Il meglio:

http://www.angelikalanger.com/GenericsFAQ/JavaGenericsFAQ.html

Autoboxing

Autoboxing/Autounboxing

```
public static void main(String args[]) {
    int dim=10;
    Pila<Integer> s=new Pila(); // s= new
Coda();
    for (int k=0;k<dim;k++){
        //Integer o=new Integer(k);
        //s.inserisci(o);
        s.inserisci(k);
    }
    for (int k=0;k<3*dim;k++) {
      //int j= Integer.parseInt(s.estrai());
      int j= s.estrai();
      System.out.println(j);
    }
}
```

Esercizio

Esercizio:

Modificare la Tombola usando i generics