Wrappers e Autoboxing

Come inserisco interi in una Collection?

```
Collection<int> coll=new ArrayList();
g=0;
tring s:a) {
teger z;
```

unexpected type
required: reference
found: int

(Alt-Enter shows hints)

Devo usare un "Wrapper": class Integer

Class Float per i float,
Class Boolean per i boolean,
Class Double per i double,
Class Char per i char...

Using wrappers

```
public class Wrappers {
    Collection<Integer> coll=new ArrayList();
    public static void main(String a[]){
        new Wrappers(a);
    }
```

```
public Wrappers(String[] a){
        for (String s:a) {
            int i=0;
            try { i=Integer.parseInt(s) }
            catch (NumberFormatException e) {
                System.out.println(s+ "non e' un intero");
                System.exit(1);
            }
            coll.add(new Integer(i));

        }
        for (Integer k:coll) {
            int x=k.intValue()+3;

            System.out.println(x);
        }; } }
```

Autoboxing and Unboxing

Autoboxing is the automatic conversion that the Java compiler makes between the primitive types and their corresponding object wrapper classes. For example, converting an int to an Integer, a double to a Double, and so on. If the conversion goes the other way, this is called unboxing.

Here is the simplest example of autoboxing:

```
Character ch = 'a';
```

Da: The Java™ Tutorials

Using autoboxing

```
public class Wrappers {
    Collection<Integer> coll=new ArrayList();
    public static void main(String a[]){
        new Wrappers(a);
    }
public Wrappers(String[] a){
        for (String s:a) {
            int i=0;
            try { i=Integer.parseInt(s) }
            catch (NumberFormatException e) {
                System.out.println(s+ "non e' un intero");
                System.exit(1);
            }
            //coll.add(new Integer(i));
            coll.add(i);
        }
        for (Integer k:coll) {
            //int x=k.intValue()+3;
            int x=k+3;
            System.out.println(x);
        }; } }
```

Un esempio riassuntivo

Esempio: Tombola!

Tombola

Il croupier(banco) ha a disposizione un tabellone sul quale sono riportati tutti i numeri da 1 a 90, e un sacchetto riempito con pezzi numerati in modo analogo.

Il suo compito consiste nell'estrarre i pezzi in modo casuale, e annunciare agli altri giocatori il numero uscito.

I giocatori dispongono di una o più cartelle precedentemente acquistate, composte da 3 righe, su ciascuna delle quali sono riportati cinque numeri compresi tra 1 e 90.

Ogni volta che il numero estratto è presente su una o più delle sue schede, il giocatore "copre" la casella corrispondente.

Lo scopo ultimo del gioco è quello di realizzare la tombola, ovvero arrivare per primi a coprire tutti i numeri presenti su una delle proprie cartelle.

Estratto da https://it.wikipedia.org/wiki/Tombola

Tombola

Il **croupier**(**banco**) ha a disposizione un **tabellone** sul quale sono riportati tutti i numeri da 1 a 90, e un **sacchetto** riempito con **pezzi** numerati in modo analogo.

Il suo compito consiste nell'**estrarre** i pezzi in modo casuale, e **annunciare** agli altri giocatori il **numero uscito**.

I **giocatori** dispongono di una o più **cartelle** precedentemente acquistate, composte da 3 **righe**, su ciascuna delle quali sono riportati cinque numeri compresi tra 1 e 90.

Ogni volta che il numero estratto è presente su una o più delle sue schede, il giocatore "**copre**" la casella corrispondente.

Lo scopo ultimo del gioco è quello di realizzare la tombola, ovvero **arrivare per primi a coprire** tutti i numeri presenti su una delle proprie cartelle.

Estratto da https://it.wikipedia.org/wiki/Tombola

Diagramma ad oggetti...

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<Integer> sacchetto;

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

NOTA:
getNextNumber(i) dà numeri tra 0 e i-1
List indexes are 0 based

Banco – unit test

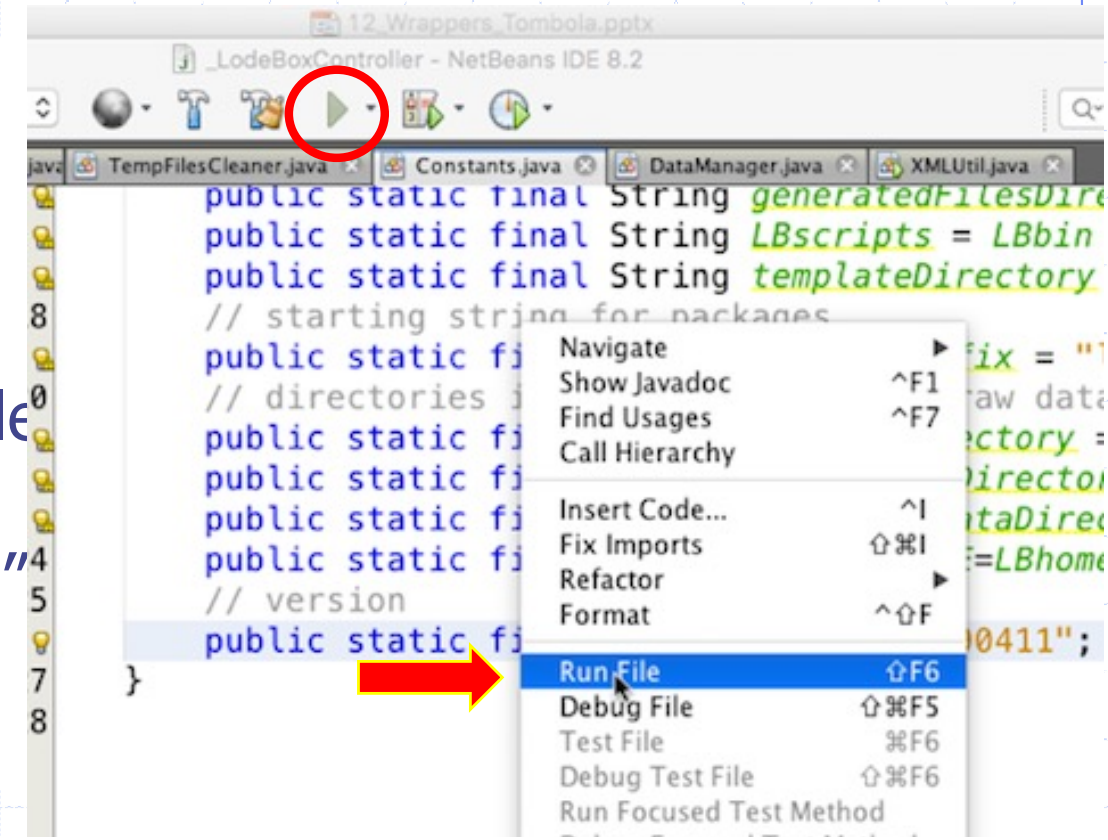
```
public static void main(String[] args) {
    Banco banco = new Banco();
    while (true) {
        banco.getNextNumber();
    }
}
}
```

Perché scriviamo questo main?

```
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
```

Si può scrivere un main per ogni classe! (unit test)

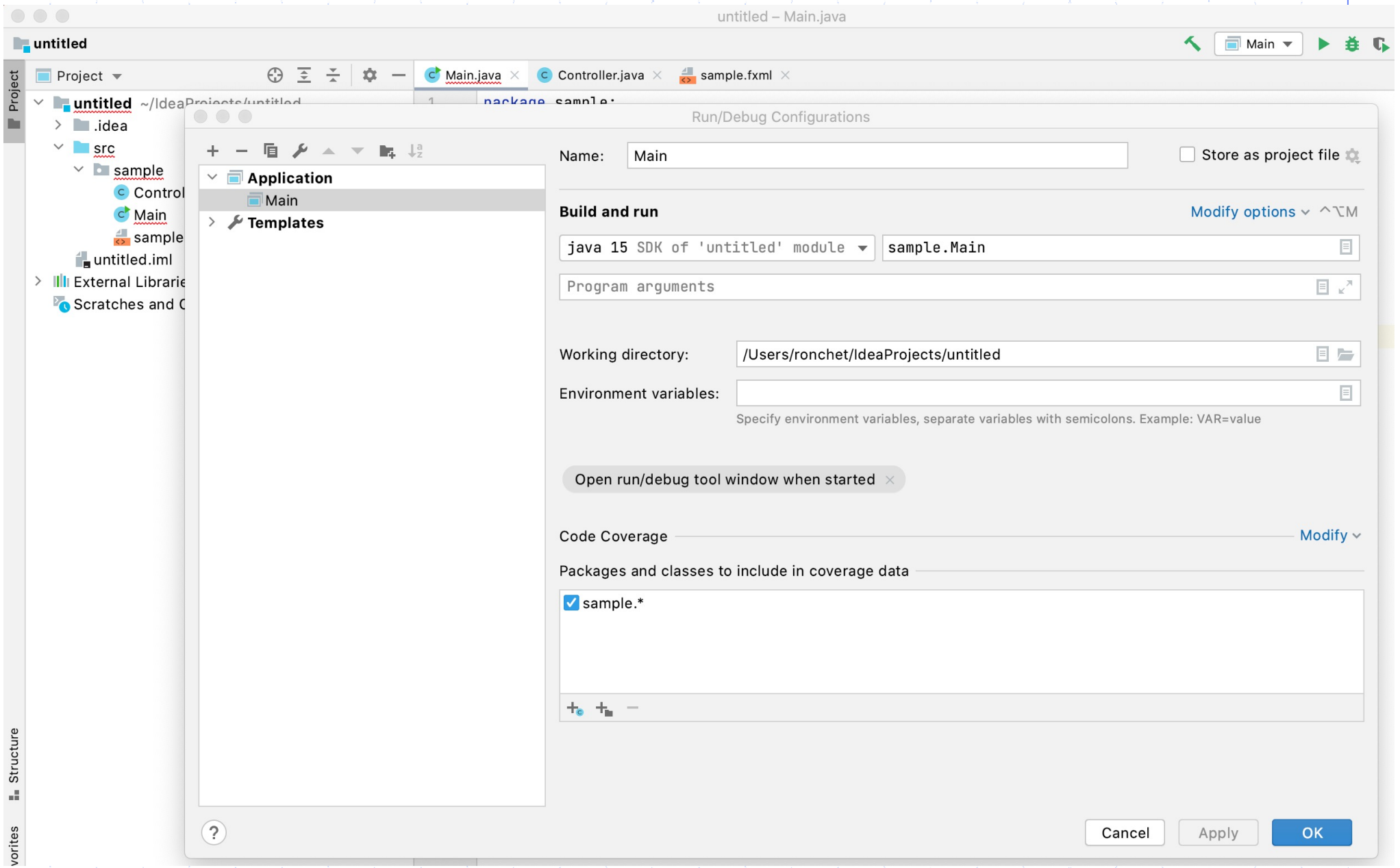
Dall'IDE, verrà lanciato quello definito tra le proprietà del progetto (triangolo verde in alto in Netbeans), o quello del file corrente se con il tasto destro scegliamo "run file" dal menu contestuale



L'IDE si occupa di scrivere il nome della classe da lanciare nel Manifest del jar.

E' possible definire più configurazioni di lancio, e di sceglire poi dal triangolino quella voluta.

Definizione della classe da lanciare in Intellij



Cartella

```
package tombola;

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

public class Cartella {
    private HashSet<Integer> numeri = new HashSet();
    private HashSet<Integer> 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(x);
                if (creatoNuovoNumero) System.out.println("aggiunto "+ x);
            } while (!creatoNuovoNumero);
        }
        mancanti.addAll(numeri);
    }
```

Cartella

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

private void print(HashSet<Integer> list) {
    Iterator<Integer> 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

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

private void print(HashSet<Integer> list) {
    for (Integer x: list) {
        System.out.print(x+" ");
    }
    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);
 }
}
```

Nota bene!

In Object Oriented Programming,

il main è l'ultima cosa che si scrive!

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
```

Tombola - esercizio

Esercizio:

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