# Classe

```
public class BankAccount {
  public BankAccount() { balance = 0; }
  public void deposit(double amount) {
     balance += amount;}
  public void  withdraw(double amount) {
     balance -= amount;}
  public double  getBalance() {
     return balance; }
  private double balance;

  public static void main(String args[]) {
     BankAccount mio = new BankAccount();
     BankAccount tuo = new BankAccount();

     mio.deposit(25.8);
     tuo.deposit(25.8);
     tuo.withdraw(11.7);
     System.out.println("mio balance = " +
        mio.getBalance());
     System.out.println("tuo balance = " +
        tuo.getBalance());
  }
}
```

# Costruttore di default

```
public class BankAccount {
  public void  deposit(double amount) {
    balance += amount;
  }
  public void  withdraw(double amount) {
    balance -= amount;
  }
  public double  getBalance() {
    return balance;
  }
  private double balance;
}

public static void main(String args[]) {

    BankAccount mio = new BankAccount();
    BankAccount tuo = new BankAccount();

    mio.deposit(25.8);
    tuo.deposit(25.8);
    tuo.withdraw(11.7);

    System.out.println("mio balance = " +
      mio.getBalance());
    System.out.println("tuo balance = " +
      tuo.getBalance());
  }
}
```

# Overloading dei costruttori

```
class BankAccount {
  public BankAccount() { balance = 0; }
  public BankAccount(double initialBalance) {
    balance = initialBalance;
   }
   public void  deposit(double amount) {
    balance += amount;
   }
   public void  withdraw(double amount) {
    balance -= amount;
   }
   public double  getBalance() {
    return balance;
   }
  private double balance;

  public static void main(String args[]) {

    BankAccount mio = new BankAccount();
    BankAccount tuo = new BankAccount(10.51);

    mio.deposit(25.8);
    tuo.deposit(25.8);
    tuo.withdraw(11.7);

    System.out.println("mio balance = " +
                                  mio.getBalance());
      System.out.println("tuo balance = " +
                                  tuo.getBalance());
  }
}
```

# Parola chiave `this` (I)

```
class BankAccount {
  public BankAccount() { balance = 0; }
  public BankAccount deposit(double amount) {
    balance += amount;
    return this;
   }
   public BankAccount  withdraw(double amount) {
    balance -= amount;
    return this;
   }
   public double  getBalance() {
    return balance;
   }
   private double balance;

  public static void main(String args[]) {
    double bal = 0;
    BankAccount mio = new BankAccount();

    bal =
      mio.deposit(10).withdraw(5).getBalance();
    System.out.println("mio balance = "+ bal);
    // 5.0
    System.out.println("mio balance  "+
      mio.getBalance()); // 5.0
  }
}
```

# Parola chiave `this` (II)

```
class BankAccount {
 public BankAccount() {
   this(0);
 }
 public BankAccount(double initialBalance) {
   balance = initialBalance;
 }
 public void  deposit(double amount) {
   balance += amount;
 }
 public void  withdraw(double amount) {
   balance -= amount;
 }
 public double  getBalance() {
   return balance;
 }
 private double balance;

  public static void main(String args[]) {
   BankAccount mio = new BankAccount();
   BankAccount tuo = new BankAccount(10.0);

   mio.deposit(5.0);
   tuo.deposit(15.0);
   tuo.withdraw(4.0);

   System.out.println("mio balance = " +
                              mio.getBalance());
   System.out.println("tuo balance = " +
                              tuo.getBalance());
 }
}
```

# Passaggio dei parametri (I-a)

```
class BankAccount {
  public static void chooseAccount(BankAccount
                  better, BankAccount candidate1,
                        BankAccount candidate2) {
    if ( candidate1.getBalance() >
         candidate2.getBalance() )
      better = candidate1;
    else
      better = candidate2;}
  public void  deposit(double amount) {
    balance += amount; }
  public void  withdraw(double amount) {
    balance -= amount; }
  public double  getBalance() {
    return balance; }
  private double balance;

  public static void main(String args[]) {

    BankAccount mio = new BankAccount(),
                tuo = new BankAccount();
    BankAccount migliore = null;

    mio.deposit(100); tuo.deposit(50);
    BankAccount.chooseAccount(migliore,mio,tuo);
    System.out.println(migliore); // null
  }
}
```

# Passaggio dei parametri (I-b)

```
class BankAccount {
  public static BankAccount chooseAccount(
                              BankAccount candidate1,
                            BankAccount candidate2) {
    BankAccount better;
    if ( candidate1.getBalance() >
         candidate2.getBalance() )
      better = candidate1;
    else
       better = candidate2;
    return better;
  }
 public void  deposit(double amount) {
    balance += amount; }
 public void  withdraw(double amount) {
    balance -= amount; }
 public double  getBalance() {
    return balance; }
 private double balance;

  public static void main(String args[]) {
    BankAccount mio = new BankAccount(),
                            tuo = new BankAccount();
    BankAccount migliore = null;

    mio.deposit(100); tuo.deposit(50);
    migliore = BankAccount.chooseAccount(mio,
                                          tuo);
    if(migliore==mio) System.out.println("mio");
    else System.out.println("mio");
  }
}
```

# Passaggio dei parametri (II)

```
class BankAccount {
  public void  deposit(double amount) {
    balance += amount; }
  public void  withdraw(double amount) {
    balance -= amount; }
  public double  getBalance() {
    return balance; }
  public void transfer(BankAccount other,
                                  double amount) {
    balance -= amount;
    other.deposit(amount); }
  private double balance;

  public static void main(String args[]) {

    BankAccount mio = new BankAccount();
    BankAccount tuo = new BankAccount();
    mio.deposit(100);
    mio.transfer(tuo, 50);
    System.out.println("mio balance  = " +
                               mio.getBalance());
    System.out.println("tuo balance  = " +
                               tuo.getBalance());
  }
}
```

# Copia di oggetti (I)

```
class BankAccount {
  public BankAccount() { balance = 0; }
  public BankAccount(double initialBalance) {
    balance = initialBalance; }
  public void deposit(double amount) {
    balance += amount; }
  public void withdraw(double amount) {
    balance -= amount; }
  public double  getBalance() {
    return balance; }
  private double balance;

  public static void main(String args[]) {
    BankAccount mio = new BankAccount();
    BankAccount b;
    b = mio;
    b.deposit(500);
    System.out.println("Account " +
                       ": balance  = " +
                       mio.getBalance());
    BankAccount tuo = new
                    BankAccount(mio.getBalance());
    System.out.println("Account " +
                       ": balance  = " +
                       tuo.getBalance());
  }
}
```

# Copia di oggetti (II)

```
class BankAccount {
  public BankAccount() { balance = 0; }
  public BankAccount(double initialBalance) {
    balance = initialBalance; }
  public void deposit(double amount) {
    balance += amount; }
  public void withdraw(double amount) {
    balance -= amount; }
  public double  getBalance() {
    return balance; }
  public void copy(BankAccount other) {
    this.balance = other.getBalance();
    // this.balance = other.balance;
  }
  private double balance;

  public static void main(String args[]) {
    BankAccount mio = new BankAccount();
    BankAccount b;
    b = mio;
    b.deposit(500);
    tuo.copy(mio);
    System.out.println("Account " +
                       ": balance  = " +
                       mio.getBalance());
    BankAccount tuo = new BankAccount();
    System.out.println("Account " +
                       ": balance  = " +
                       tuo.getBalance());
  }
}
```

# Confronto di oggetti

```
class BankAccount {
 public BankAccount(double amount) {
   balance = amount;}
 public BankAccount() { this(0.0);}
 public void  deposit(double amount) {
   balance += amount; }
 public void  withdraw(double amount) {
   balance -= amount; }
 public double  getBalance() {
   return balance; }
 public boolean compareBalance(BankAccount
                                        other) {
   return getBalance() == other.getBalance();}
 private double balance;

 public static void main(String args[]) {

   BankAccount mio = new BankAccount(100),
          tuo = new BankAccount(100);
   System.out.println(mio.compareBalance(tuo));
// true
   System.out.println(mio == tuo); // false

  }
}
```

# Metodi statici

```
public class MetodiStatic {
  public static boolean approxEqual
                          (double x, double y) {
    final double EPSILON = 1E-14;
    double max = Math.max(Math.abs(x),
                                  Math.abs(y));
    return Math.abs(x-y) <= EPSILON * max;
  }

  public  static void main(String args[]) {
    System.out.println(approxEqual(1.0,
                                       1.0001))
  }
}
```

# Classe Math

```
static double abs(double a)
static float abs(float a)
static int abs(int a)
static long abs(long a)

static double cos(double a)
static double sin(double a)
static double tan (double a)

static double exp(double a)
static double log(double a)
static double pow(double a, double b)

static double sqrt(double a)

public static final double E
public static final double PI
```

# Variabili statiche

```
class BankAccount {
  public BankAccount() {this(0);}
  public BankAccount(double initialBalance) {
    accountNumber = newAccountNumber++;
    balance = initialBalance;
  }
  public void deposit(double amount) {
    balance += amount;
  }
  public void withdraw(double amount) {
    balance -= amount;
  }
 public double  getBalance() {
    return balance;
 }
 public int getAccountNumber() {
    return accountNumber;
 }
 private double balance;
 private final int accountNumber;
 private static int newAccountNumber = 1;

  public static void main(String args[]) {
    BankAccount mio = new BankAccount(); // 1
    BankAccount tuo = new BankAccount(); // 2
    System.out.println("mio account number  = "
                       + mio.getAccountNumber());
    System.out.println("tuo account number  = "
                       + tuo.getAccountNumber());
   }
}
```

# Copia di oggetti

```
class BankAccount {
  public BankAccount() { this(0); }
  public BankAccount(double initialBalance) {
    accountNumber = newAccountNumber++;
    balance = initialBalance; }
  public void deposit(double amount) {
    balance += amount; }
  public void withdraw(double amount) {
    balance -= amount; }
  public double  getBalance() {
    return balance; }
  public int getAccountNumber() {
    return accountNumber; }
  private double balance;
  private int accountNumber;
  private static int newAccountNumber = 1;

  public static void main(String args[]) {
    BankAccount mio = new BankAccount();
    BankAccount b;
    b = mio;
    b.deposit(500);
    System.out.println("Account " +
                       mio.getAccountNumber() +
            ": balance  = " + mio.getBalance());
    BankAccount tuo = new
                    BankAccount(mio.getBalance());
    System.out.println("Account " +
                       tuo.getAccountNumber() +
            ": balance  = " + tuo.getBalance());
  }
}
```

# Confronto di oggetti

```
class BankAccount {
 public BankAccount(double amount) {
   balance = amount;}
 public BankAccount() { this(0.0);}
 public void  deposit(double amount) {
   balance += amount; }
 public void  withdraw(double amount) {
   balance -= amount; }
 public double  getBalance() {
   return balance; }
 public boolean compareBalance(BankAccount
                                          other) {
   return getBalance() == other.getBalance();}
 private double balance;

 public static void main(String args[]) {

   BankAccount mio = new BankAccount(100),
         tuo = new BankAccount(100);
   System.out.println(mio.compareBalance(tuo
                                              ));
 }
}
```

# Metodo finalize

```
public class GCDemo {

  public GCDemo() {}

  public static void main(String[] args) {
    GCDemo a = new GCDemo();
    a = new GCDemo();
    System.gc(); // finalize()
  }

  protected void finalize() {
    System.out.println("finalize()");
  }
}
```

# Overloading dei metodi (I)

```
// overloading con i tipi primitivi

public class PrimitiveOverloading {
  void f1(byte x) {
    System.out.println("f1(byte)"); }
  void f1(int x) {
    System.out.println("f1(int)"); }

  void f2(int x) {
    System.out.println("f2(int)"); }

   void testConstVal() {
     System.out.println("Testing with 5");
     f1(5);f2(5);
   }
  void testByte() {
     byte x = 0;
     System.out.println("byte argument:");
     f1(x);f2(x);
   }
  public static void main(String[] args) {
     PrimitiveOverloading p =
       new PrimitiveOverloading();
     p.testConstVal(); // f1(int) f1(int)
     p.testByte(); // f1(byte) f2(int)
  }
}
```

# Overloading dei metodi (II-a)

```
// overloading con i tipi primitivi

public class PrimitiveOverloading {
 void f4(int x) {
   System.out.println("f4(int)"); }
 void f4(long x) {
   System.out.println("f4(long)"); }
 void f4(float x) {
   System.out.println("f4(float)"); }
  void f4(double x) {
   System.out.println("f4(double)"); }

  void f5(long x) {
   System.out.println("f5(long)"); }
  void f5(float x) {
   System.out.println("f5(float)"); }
  void f5(double x) {
   System.out.println("f5(double)"); }

  void f6(float x) {
   System.out.println("f6(float)"); }
  void f6(double x) {
   System.out.println("f6(double)"); }

  void f7(double x) {
   System.out.println("f7(double)"); }

 void testInt() {
    int x = 0;
    System.out.println("int argument:");
    f4(x);f5(x);f6(x);f7(x);
  }

// continua nel lucido successivo
```

# Overloading dei metodi (II-b)

```
// continua dal precedente

public static void main(String[] args) {
    PrimitiveOverloading p =
      new PrimitiveOverloading();
    p.testInt(); //
  }
}

//  f4(int) f5(long) f6(float) f7(double)
```

# Overloading dei metodi (III)

```
public class Demotion {
   void f1(double x) {
      System.out.println("f1(double)"); }

  void f2(long x) {
      System.out.println("f2(long)"); }
  void f2(float x) {
      System.out.println("f2(float)"); }

  void testDouble() {
     double x = 0;
     System.out.println("double
                              argument:");

      f1(x);f2((float)x);f2((long)x)
   }
   public static void main(String[] args) {
     Demotion p = new Demotion();
     p.testDouble();
   }
}

// f1(double) f2(float) f2(long)
```

# Array

*ArrayType:*
  *Type* `[ ]`

*ArrayCreationExpression:*
  **`new`** *PrimitiveType DimExprs* $Dims_{opt}$
  **`new`** *TypeName DimExprs* $Dims_{opt}$
  **`new`** *PrimitiveType Dims ArrayInitializer*
  **`new`** *TypeName Dims ArrayInitializer*

*DimExprs:*
  *DimExpr*
  *DimExprs DimExpr*

*DimExpr:*
`[` *Expression* `]`

*Dims:*
`[ ]`
*Dims* `[ ]`

*ArrayInitializer:*
  `{` $VariableInitializers_{opt}$ $,_{opt}$ `}`

*VariableInitializers:*
  *VariableInitializer*
  *VariableInitializers* `,` *VariableInitializer*

*VariableInitializer:*
  *Expression*
  *ArrayInitializer*

# Array

```
public class Arrays {
  public static void main(String[] args) {
    int[] a1 = { 1, 2, 3, 4, 5 };
    int[] a2;
    a2 = a1;
    for(int i = 0; i < a2.length; i++)
      a2[i]++;
    for(int i = 0; i < a1.length; i++)
     System.out.println(
              "a1[" + i + "] = " + a1[i]);
  }
}
```

# Array 2

```
public class Arrays2 {
  public static void main(String[] args) {
    int[] a1 = { 1, 2, 3, 4, 5 };
    int[] a2 = { 6, 7, 8, 9 };
    int[] a3;

    a3 = a1;
    for(int i = 0; i < a3.length; i++)
      System.out.println(
              "a3[" + i + "] = " + a3[i]);
    a3 = a2;
    for(int i = 0; i < a3.length; i++)
      System.out.println(
              "a3[" + i + "] = " + a3[i]);
  }
}
```

# Array 3

```
import java.util.*;
public class ArrayNew {
 static Random rand = new Random();
 public static void main(String[] args) {
   int[] a;
   a = new int[rand.nextInt(20)];
   System.out.println("length of a = " +
                                        a.length);
   for(int i = 0; i < a.length; i++)
      System.out.println("a[" + i + "] = "
                                          + a[i]);
   for(int i = 0; i < a.length; i++)
      a[i] = i;
 }
}
```

# Array 4

```
import java.util.*;

public class ArrayClassObj {
  static Random rand = new Random();
  public static void main(String[] args) {
    Integer[] a =
                    new Integer[rand.nextInt(20)];
    System.out.println("length of a = " +
                                        a.length);
    for(int i = 0; i < a.length; i++) {
      a[i] = new Integer(rand.nextInt(500));
      System.out.println("a[" + i + "] = " +
                                          a[i]);
    }
  }
}
```

# Array 5

```
public class ArrayInit {
  public static void main(String[] args) {
    Integer[] a = {
    new Integer(1),
      new Integer(2),
      new Integer(3),
    };
    Integer[] b = new Integer[] {
    new Integer(1),
      new Integer(2),
      new Integer(3),
    };
  }
}
```

# Classe Arrays

```
public class EsempioArrays {
  public static void main(String[] args) {
    final int MAX_ELEM_NUM = 10;
    final int MAX_ELEM_VAL = 50;

    Random r = new Random();
    int[] vett =
          new int[r.nextInt(MAX_ELEM_NUM)];
    for (int i = 0; i < vett.length; i++)
      vett[i] = r.nextInt(MAX_ELEM_VAL);
    Arrays.sort(vett);

    int k = r.nextInt(MAX_ELEM_VAL);
    ix = Arrays.binarySearch(vett, k);
    if (ix < 0) System.out.println(
                                "Value "+
                                k +
                                " not found.");
    else System.out.println("Value " + k +
                    " found at index " + ix);

  }
}
```

# Ragged Arrays

```
public class RaggedArray {
  public static void main(String[] args) {
    int[][] a = {{1},
                 {2, 3},
                 {4, 5, 6},
                 {7, 8, 9, 0}};

    for (int i = 0; i < a.length; i++) {
      for (int j=0; j < a[i].length; j++)
        System.out.print(a[i][j] + " ");
      System.out.println();
    }
  }
}

// Output
// 1
// 2 3
// 4 5 6
// 7 8 9 0
```

# Triangolo di Tartaglia

```
public class Tartaglia {
  public static void main(String[] args) {
    final int DEPTH = 10;
    int[][] t = new int[DEPTH][];

    t[0] = new int[]{1};
    t[1] = new int[]{1, 1};
    for(int i = 2; i < t.length; i++) {
    t[i] = new int[i+1];
      for (int j=0; j < t[i].length; j++){
        int a, b;
        a = (j >= t[i-1].length) ?
                                  0:t[i-1][j];
        b = (j <= 0)? 0:t[i-1][j-1];
        t[i][j] = a+b;
      }
    }
    for (int i=0; i < t.length; i++) {
      System.out.print(i + ":\t");
     for(int j=0; j<t[i].length; j++)
       System.out.print(t[i][j] + " ");
     System.out.println("");
    }
  }
}
```

# La classe Character

```
public class CharacterDemo {
  public static void main(String[] args) {
    Character a = new Character('a');
    Character a2 = new Character('a');
    Character b = new Character('b');

    int difference = a.compareTo(b);

    if (difference == 0) {
      System.out.println("a is equal to b.");
    } else if (difference < 0) {
      System.out.println("a is less than b.");
    } else if (difference > 0) {
      System.out.println("a is greater than
                          b.");
    }
    System.out.println("a is " +
                       ((a.equals(a2)) ?
                        "equal" : "not equal") +
                        " to a2.");
    System.out.println("The character " +
                        a.toString() + " is " +
          (Character.isUpperCase(a.charValue()) ?
                     "upper" : "lower") +
                              "case.");
  }
}

// Output
// a is less than b.
// a is equal to a2.
// The character a is lowercase.
```

# Argomenti del main

```
public class ArgsDemo {

  public static void main(String[] args) {
    for (int i = 0; i < args.length; i++)
      System.out.print(args[i]);
    System.out.println();
  }
}
```

# Stray pointer I

```
int myStrCopy(char *dest, char *src)
{
    for ( ; *src != '\0'; src++, dest++)
        *dest = *src;
}

main() {
    char *s = "HotJava is Cool!";
    char t[] = "Java is Cool!";

    printf("%s, %s\n", s, t);
    myStrCopy(t, s);
    printf("%s, %s\n", s, t);
}

/* Output
HotJava is Cool!, HotJava is Cool!%s, %s
*/
```

# Stray pointer II

```
public class StrCpy {
  public static void main(String[] args) {
    String s = "HotJava is Cool!";
    StringBuffer t =
         new StringBuffer("Java is Cool!");
    System.out.println(s + ", " + t);
    myStrCopy(t, s);
  System.out.println(s + ", " + t);
  }

  public static void myStrCopy(
       StringBuffer dest, String src) {
  int i, len = src.length();

  for (i = 0; i < len; i++)
    dest.setCharAt(i, src.charAt(i));
  }
}
```