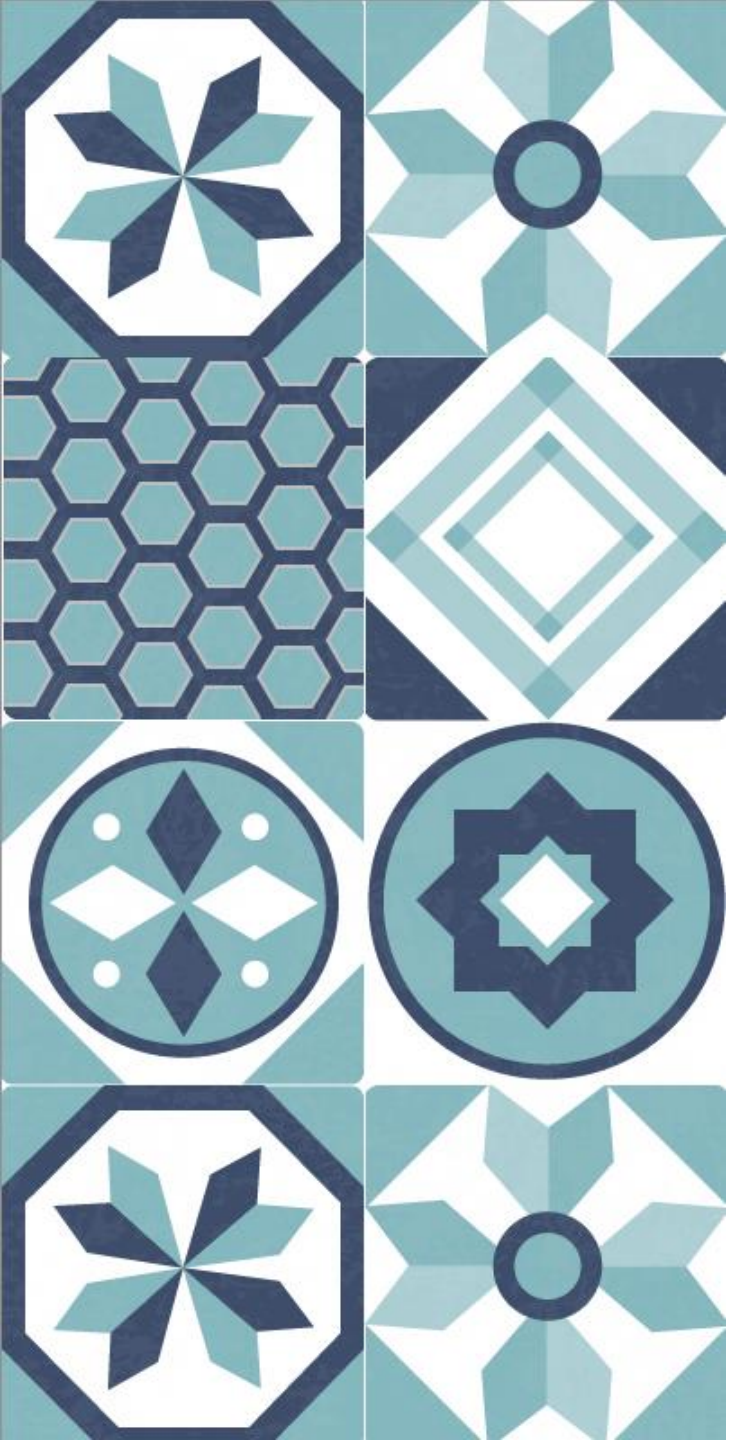


FONDAMENTI DI INFORMATICA

Alma Artis
Francesca Pratesi (ISTI, CNR)

Processing – Oggetti, Array

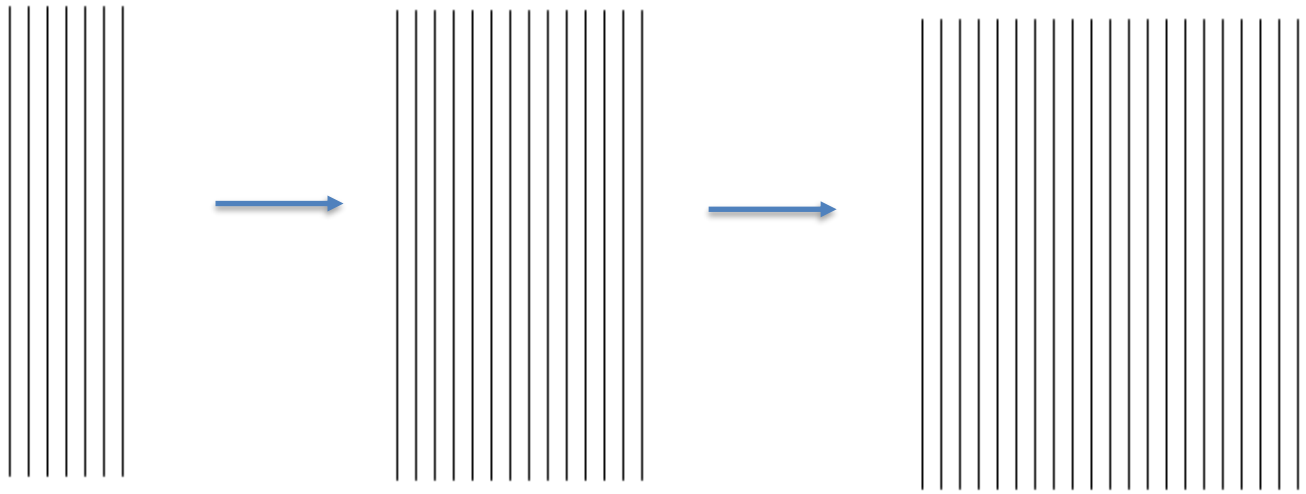
FERMARE L'ESECUZIONE

METTERE IN PAUSA L'ESECUZIONE

- In alcuni casi può essere utile mettere in pausa l'esecuzione del vostro programma
- Questo può essere fatto con la funzione
- `delay(time);`
- Dove time è il tempo espresso in millisecondi

ESEMPIO: LINEE PARALLALE

```
int x = 10;
int y = 100;
int len = 250;
int spacing = 10;

void setup(){
  size(400,400);
  background(255);
}

void draw(){
 for(int i=0; i<7; i++){
    line(x,y,x,y+len);
    x = x + spacing;
  }
  delay(5000);
}
```



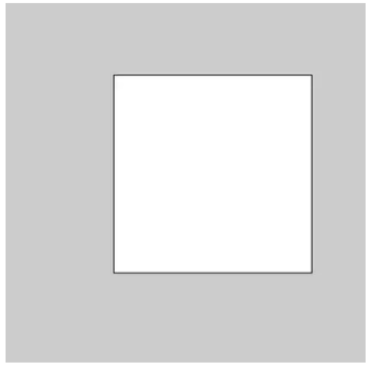
PERSONALIZZAZIONE

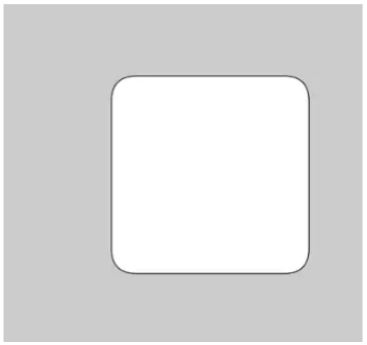
PERSONALIZZARE UN RETTANGOLO

- È possibile disegnare anche rettangoli con angoli arrotondati
- Basta aggiungere parametri alla primitiva
- Un quinto parametro rappresenterà quanto arrotondare gli angoli
- È possible dare valori diversi ai quattro angoli, in questo caso saranno necessari anche un sesto, settimo e ottavo parametro

PERSONALIZZARE UN RETTANGOLO - ESEMPIO

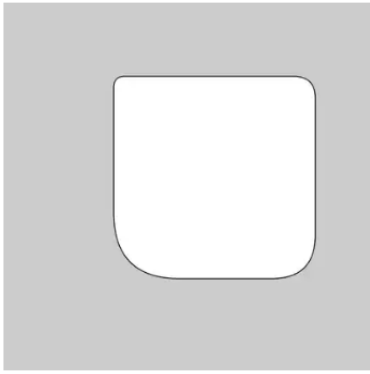
```
rect(120, 80, 220, 220);
```





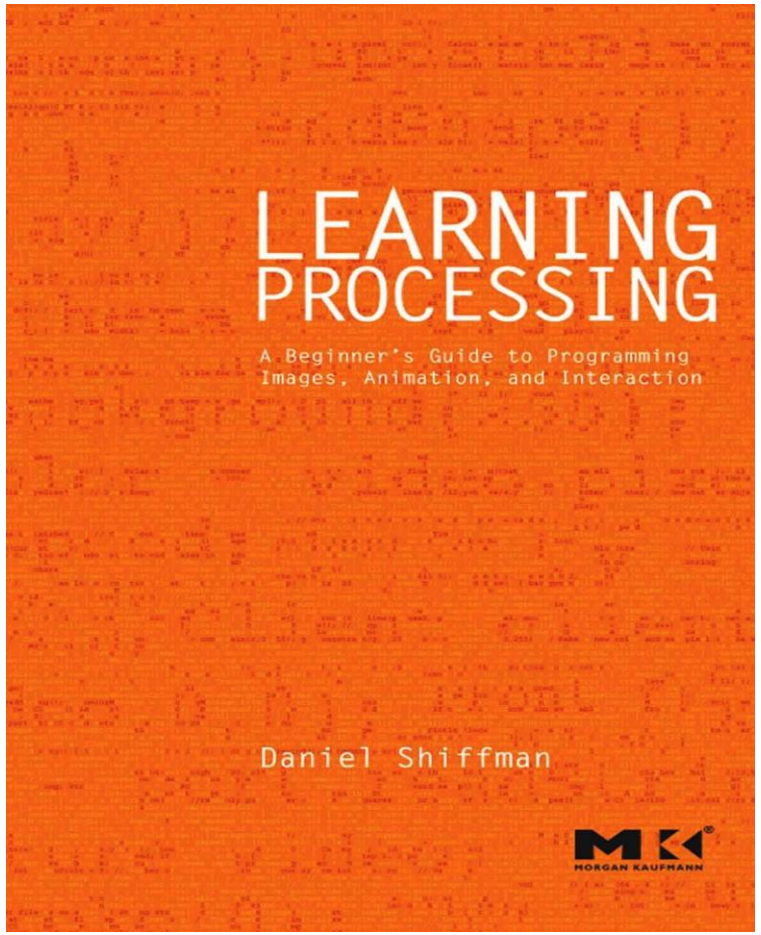
```
rect(120, 80, 220, 220, 28);
```

```
rect(120, 80, 220, 220, 12, 24, 48, 72);
```



OGGETTI

LIBRI E RIFERIMENTI



- Capitolo 8

Learning Processing– Second Edition
Daniel Shiffman
Available here: http://learningprocessing.com/

OGGETTI

- Fino abbiamo visto:
  - Variabili, condizionali, cicli, funzioni
- La programmazione ad oggetti mette insieme tutti questi elementi
- Cambia il modo di strutturare gli algoritmi che implementiamo
- Esempio (Essere Umano)
  - Dati
    - Altezza, peso, colore occhi, colore dei capelli, gender
  - Funzioni
    - Sveglia, Alzati, Mangia, Cammina, ecc.
- Distinzione tra Classe e Oggetto

ESERCIZIO SKETCH AUTO (FUNZIONI)

- Dati (variabili globali)
  - Colore, posizione x, posizione y, velocità
- Setup
  - Scegli un colore
  - Scegli una posizione
  - Scegli una velocità
- Draw
  - Riempi lo sfondo
  - Disegna la macchina alla posizione
  - Modifica la posizione in base alla velocità

AUTO CON FUNZIONI

```
color c;
float xpos;
float ypos;
float xspeed;

void setup(){
  size(400,400);
  c=color(255);
  xpos=width/2;
  ypos=height/2;
  xspeed=1;
}

void draw(){
 background(0);
 display();
 drive();
}
```

```
void display(){
  rectMode(CENTER);
  fill(c);
  rect(xpos,ypos,20,10);
}

void drive(){
  xpos=xpos+xspeed;
  if(xpos > width) xpos = 0;
}
```

CLASSI

- Se l'oggetto che stiamo modellando ha delle caratteristiche o delle funzionalità che vorremmo astrarre
  - (per esempio perché è più comodo sul momento…
  - o perché prevediamo di riusarlo il futuro)
- allora può convenire pensare di creare una classe per quell'oggetto
- Esempio: vogliamo definire una serie di automobili, ognuna che si muove orizzontalmente nello spazio, ma ad altezze diverse della finestra

CLASSI: DI CHE COSA HO BISOGNO?

- Dati
  - Le caratteristiche dell'oggetto
- Costruttore
  - Per creare l'oggetto
  - Qua inizializzerò i suoi dati
- Funzionalità
  - Le funzioni che potrò usare per quell'oggetto

CREAZIONE DI UNA CLASSE

```
// Simple non OOP Car

color c;
float xpos;
float ypos
float xspeed;

void setup() {
  size(200, 200);
  c = color(255);
  xpos = width/2;
  ypos = height/2;
  xspeed = 1;
}

void draw () {
  background(0);
  display();
  drive();
}

void display () {
  rectMode(CENTER);
  fill(c);
  rect(xpos, ypos, 20, 10);
}

void drive() {
  xpos = xpos + xspeed;
  if (xpos > width) {
    xpos = 0;
  }
}
```

```
class Car {

  color c;
  float xpos;
  float ypos;
  float xspeed;

  Car() {
    c = color(255);
    xpos = width/2;
    ypos = height/2;
    xspeed = 1;
  }

  void display () {
    rectMode(CENTER);
    fill(c);
    rect(xpos, ypos, 20, 10);
  }

  void drive() {
    xpos = xpos + xspeed;
    if (xpos > width) {
      xpos = 0;
    }
  }
```

The class name

Data

Constructor

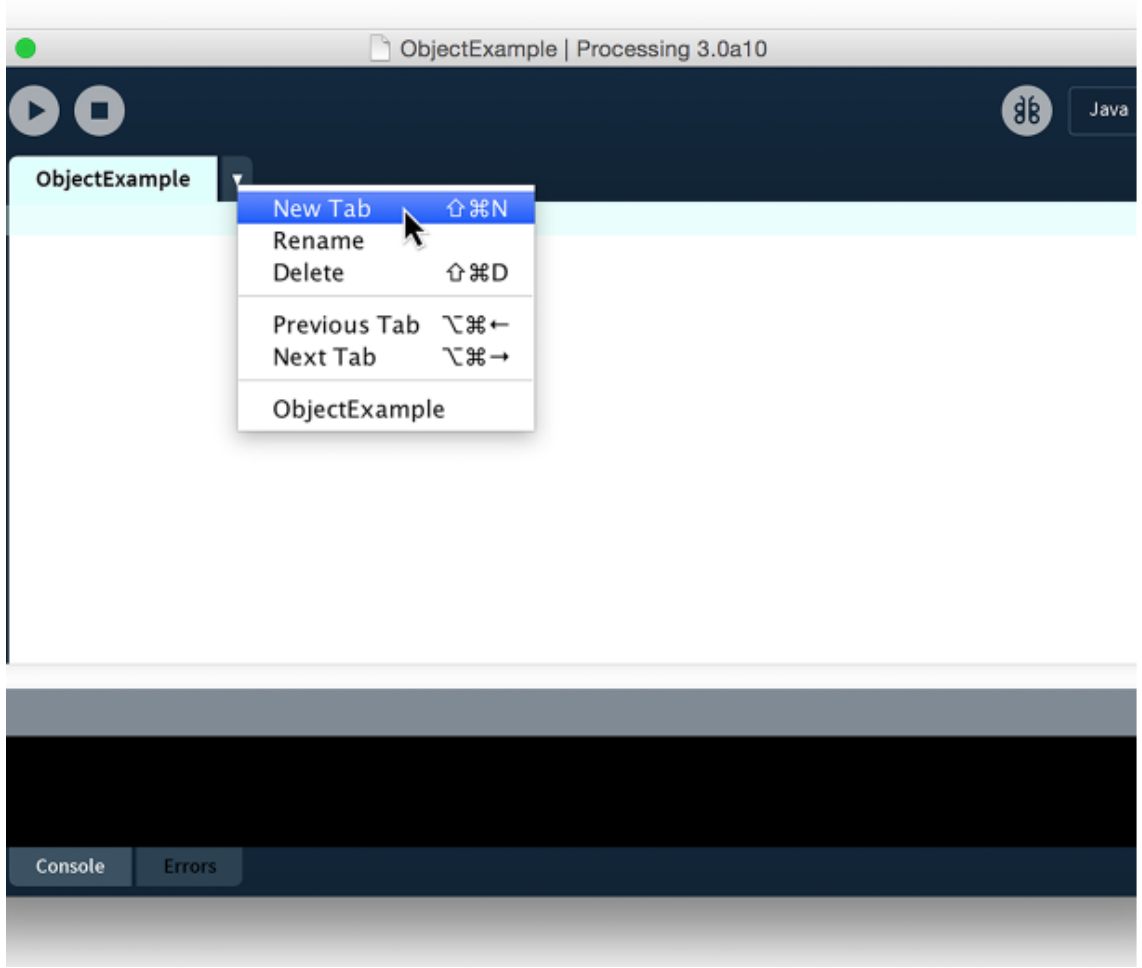
Functionality

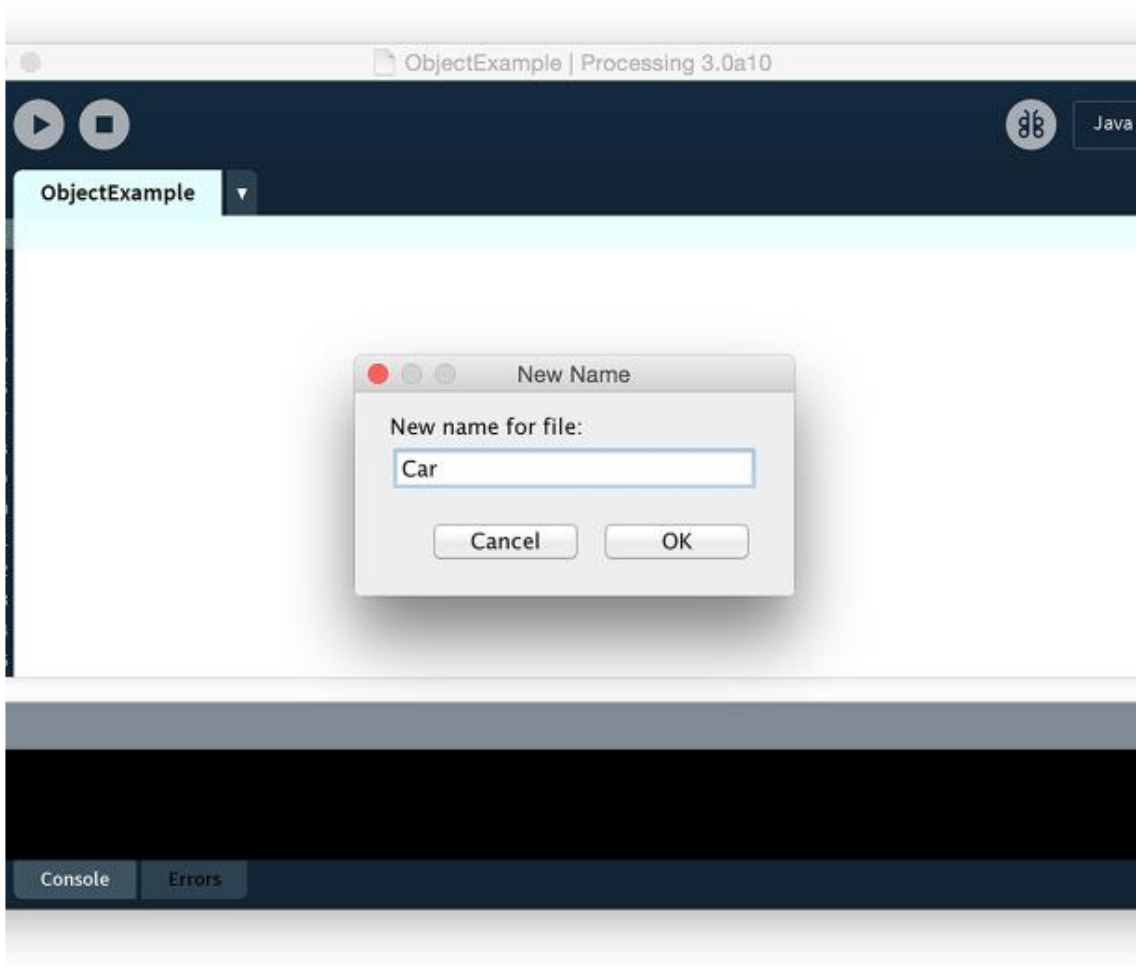
CREAZIONE DI UNA CLASSE

A class is a new block of code!

```
void setup() {

}

void draw() {

}

class Car {

}
```

CREAZIONE DI UNA CLASSE





AUTO CON CLASSI: DICHIARAZIONE

```
class Car{
  color c;
  float xpos;
  float ypos;
  float xspeed;

  Car(){
    c=color(255);
    xpos=width/2;
    ypos=height/2;
    xspeed=1;
  }

  void display(){
    rectMode(CENTER);
    fill(c);
    rect(xpos,ypos,20,10);
  }

  void drive(){
    xpos=xpos+xspeed;
    if(xpos > width) xpos = 0;
  }
}
```

USO DI UNA CLASSE

```
Car auto;

void setup(){
  size(400,400);
  auto = new Car();
}

void draw(){
  background(0);
  auto.drive();
  auto.display();
}
```

UTILITÀ DELLE CLASSI

- È possibile definire un comportamento generale di un oggetto
- E avere diverse **istanze** di quell'oggetto
  - Ogni istanza si comporterà come un oggetto standard
  - Ma sarà indipendente dalle altre istanze
  - Potrà avere quindi delle proprie caratteristiche, e un proprio comportamento

ISTANZE MULTIPLE - ESEMPIO

```
class Car{
  color c;
  float xpos;
  float ypos;
  float xspeed;

  Car(){
    c=color(random(150,255));
    xpos=width/2;
    ypos=height/(random(2,7));
    xspeed=random(1,5);
  }

  void display(){
    rectMode(CENTER);
    fill(c);
    rect(xpos,ypos,20,10);
  }

  void drive(){
    xpos=xpos+xspeed;
    if(xpos > width) xpos = 0;
  }
}
```

```
Car c1;
Car c2;
Car c3;

void setup(){
  size(400,400);
  c1 = new Car();
  c2 = new Car();
  c3 = new Car();
}

void draw(){
  background(0);
  c1.display();
  c1.drive();
  c2.display();
  c2.drive();
  c3.display();
  c3.drive();
}
```

COSTRUTTORE

- Dall'esempio precedente posso creare diverse istanze della classe Car chiamando ripetutamente l'istruzione new
- Tutte le istanze create saranno identiche
- Per modificare le proprietà di ogni oggetto, possiamo passare degli argomenti al costruttore
  - Car myCar = new Car(color(255,0,0), 0, 100, 2);

UTILITÀ DEL COSTRUTTORE

- Con il costruttore siamo noi ad impostare le caratteristiche salienti che vogliamo attribuire ad ogni istanza della classe

PASSAGGIO DI PARAMETRI AL COSTRUTTORE

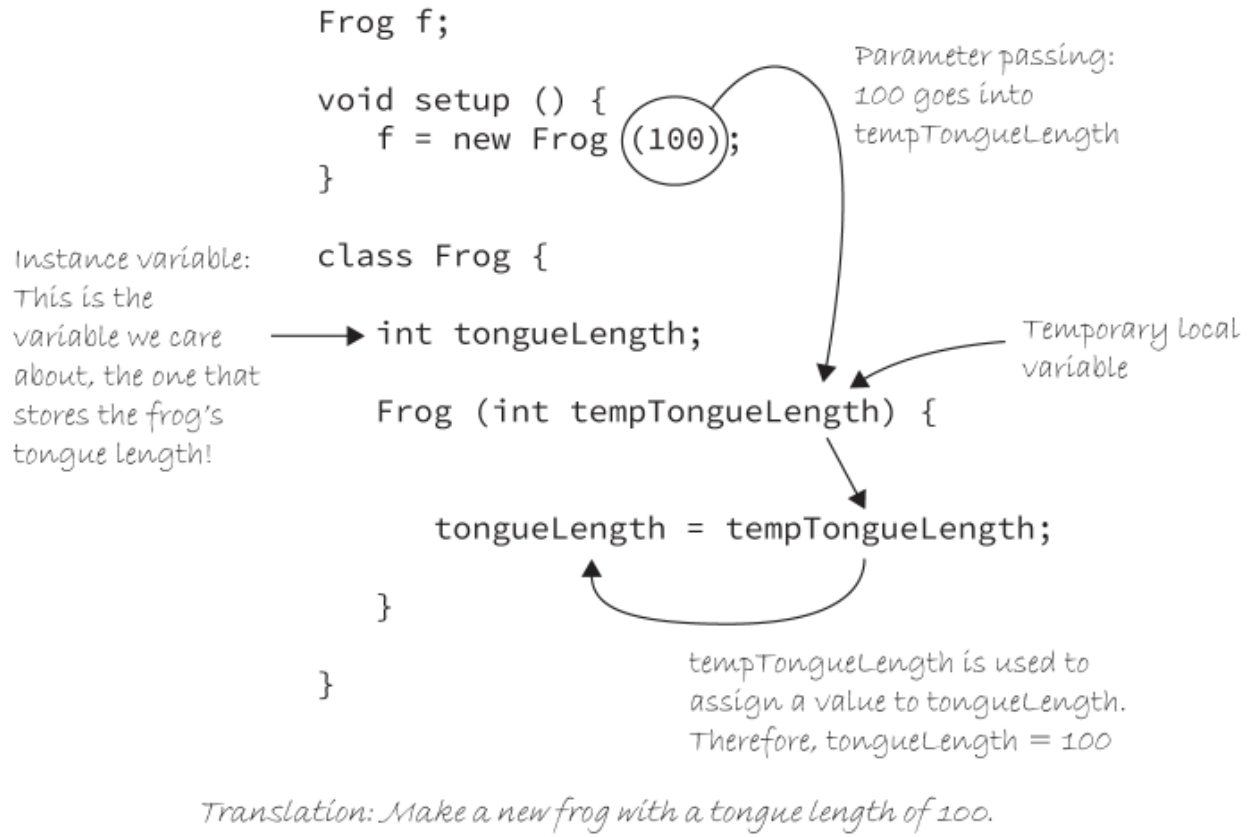


Figure 8-6

COSTRUTTORE - ESEMPIO

```
Car c1;
Car c2;
Car c3;

void setup(){
  size(400,400);
  int c = color(random(150,255));
  int h = height/4;
  int s = (int) random(1,5);
  c1 = new Car(c,h,s);
  c = color(random(150,255));   h = height/2;
  s = (int) random(1,5);
  c2 = new Car(c,h,s);
  c = color(random(150,255));  h = height/4*3;
  s = (int) random(1,5);
  c3 = new Car(c,h,s);
}

void draw(){
  background(0);
  c1.display();   c1.drive();
  c2.display();   c2.drive();
  c3.display();   c3.drive();
}
```

```
class Car{
  color c;
  float xpos;
  float ypos;
  float xspeed;

  Car(int colore, int altezza, int velocita){
    c=colore;
    xpos=width/2;
    ypos=altezza;
    xspeed=velocita;
  }

  void display(){
    rectMode(CENTER);
    fill(c);
    rect(xpos,ypos,20,10);
  }

  void drive(){
    xpos=xpos+xspeed;
    if(xpos > width) xpos = 0;
  }
}
```

ESERCIZIO

- Data la classe Car (riportata a destra), definire le opportune funzioni setup e draw per creare 3-4 istanze di Car
- Non affidatevi al random, ma scegliete voi i valori con cui istanziare ogni oggetto
  - Per esempio, la prima auto sarà rossa e più veloce delle altre, la seconda auto sarà blu e la più lenta

```
class Car{
  color c;
  float xpos;
  float ypos;
  float xspeed;

  Car(int colore, int altezza, int velocita){
    c=colore;
    xpos=width/2;
    ypos=altezza;
    xspeed=velocita;
  }

  void display(){
    rectMode(CENTER);
    fill(c);
    rect(xpos,ypos,20,10);
  }

  void drive(){
    xpos=xpos+xspeed;
    if(xpos > width) xpos = 0;
  }
}
```

ARRAY

RIASSUNTO DEI PASSI NECESSARI PER UN ARRAY

- Dichiarazione
- Creazione
- Inizializzazione
- Indicizzazione/modifica/uso

ARRAY

- Un array rappresenta una lista di variabili che hanno un nome comune
- E' possibile utilizzare tante variabili senza creare un nome diverso per ognuna
  - Esempio: `ball1, ball2, ball3`
- Quando è necessario gestire tanti elementi dello stesso tipo, un array è una soluzione molto efficace e compatta
- Ogni array è composto da una lista di elementi; ogni elemento ha un indice che identifica la posizione dell'elemento nell'array

ARRAY

- Una variabile rappresenta un nome per una locazione in memoria
- Allo stesso modo, un array è un nome che rappresenta una lista di posizioni contigue in memoria
- Un array mantiene l'ordine degli elementi inseriti

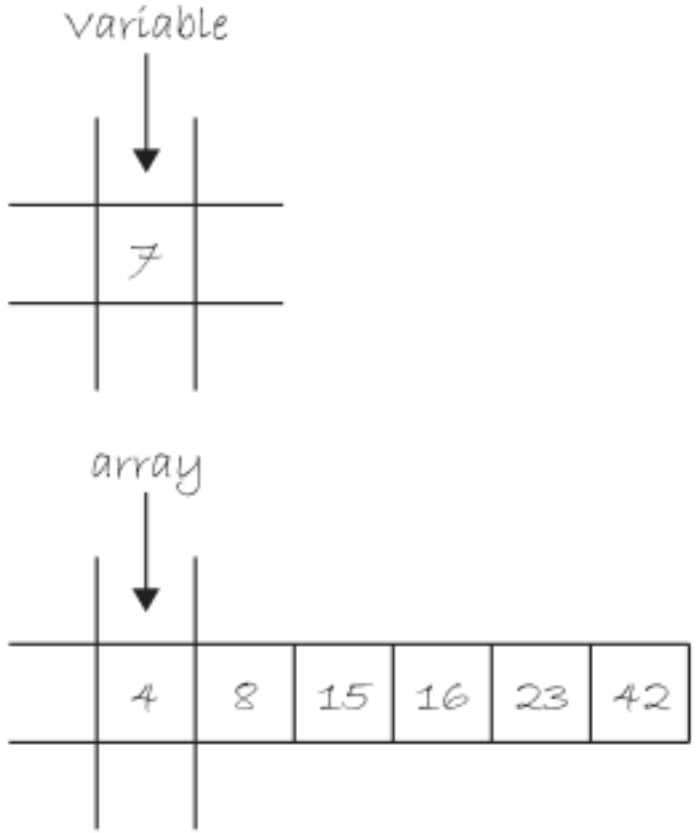


Figure 9-1

POSIZIONE DEGLI ELEMENTI

- Ogni elemento ha un indice all'interno della lista
- Il primo elemento è nella posizione 0 (zero)
  - Una distanza zero dall'inizio

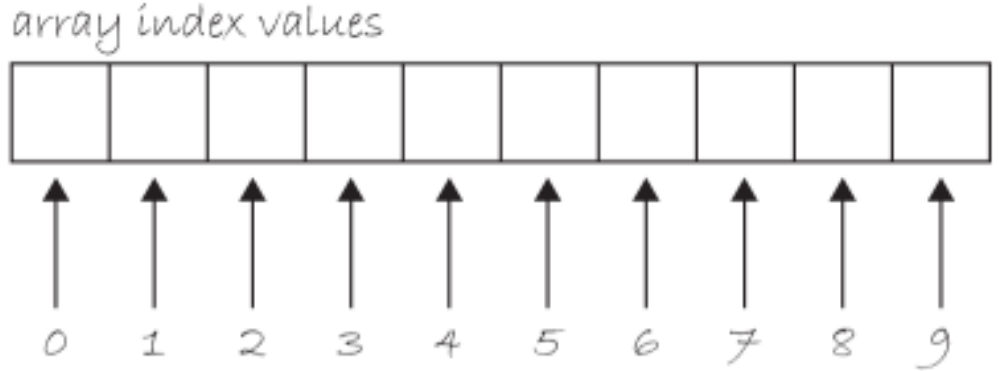


Figure 9-2

DICHIARAZIONE DI UN ARRAY

- La creazione di un array è simile alla dichiarazione di una nuova variabile: si usano le parentesi quadre per identificare il tipo della variabile
- Al momento della dichiarazione dell'array non è necessario specificare la lunghezza
- Una volta definita la lunghezza, questa non può più essere modificata

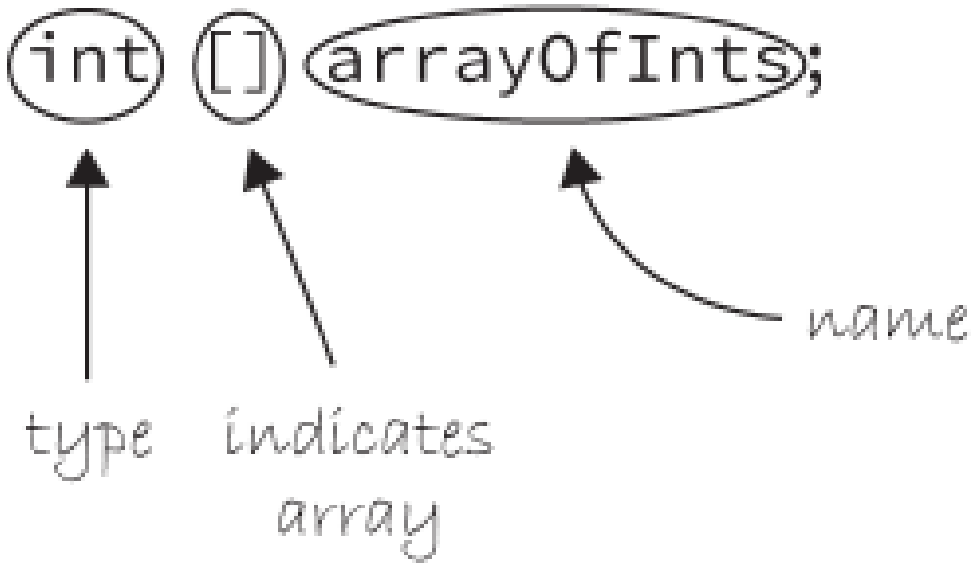


Figure 9-3

CREAZIONE DI UN ARRAY

- Per utilizzare un array appena dichiarato è necessario creare lo spazio necessario
- Bisogna decidere quante locazioni deve contenere
- Si usa di nuovo la parola chiave new come per gli oggetti

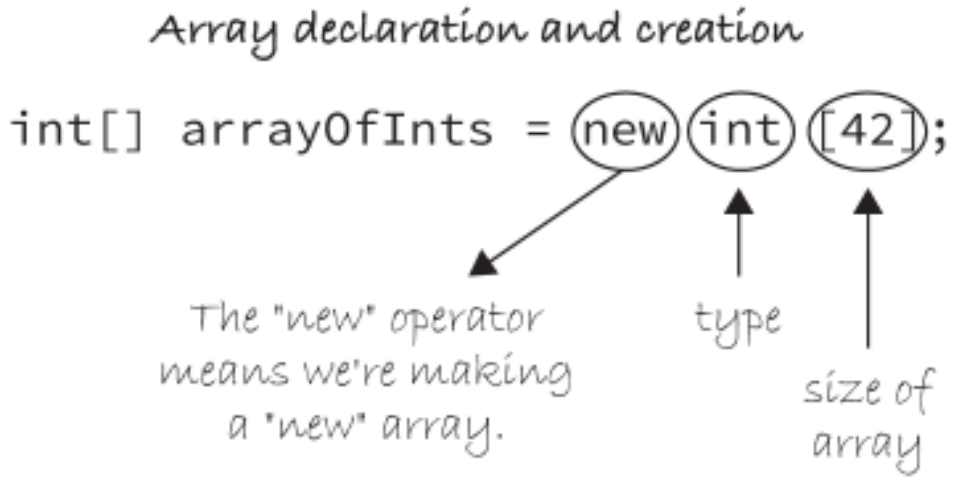


Figure 9-4

ESEMPI DI ARRAY

Example 9-1. Additional array declaration and creation examples

```
float[] scores = new float[4];                    // A list of 4 floating point numbers
Human[] people = new Human[100];                  // A list of 100 Human objects
int num = 50;
Car[] cars = new Car[num];                        // Using a variable to specify size
Spaceship[] ships = new Shapeship[num*2 + 3];     // Using an expression to
                                                      specify size
```

INIZIALIZZAZIONE DI UN ARRAY (1)

- Dopo la creazione dell'array abbiamo a disposizione un numero di locazioni fissato
- Il contenuto di ogni locazione è al momento non definito
- Bisogna inizializzare l'array con i valori iniziali che l'array dovrà contenere
- Abbiamo due modi. Inizializzazione di ogni singola posizione:

Example 9-2. Initializing the elements of an array one at a time

```
int[] stuff = new int[3];

stuff[0] = 8; // The first element of the array equals 8
stuff[1] = 3; // The second element of the array equals 3
stuff[2] = 1; // The third element of the array equals 1
```

INIZIALIZZAZIONE DI UN ARRAY

- Secondo metodo: lista di valori tra parentesi graffe

Example 9-3. Initializing the elements of an array all at once

```
int[] arrayOfInts = { 1, 5, 8, 9, 4, 5 } ;
float[] floatArray = { 1.2, 3.5, 2.0, 3.4123, 9.9 } ;
```

ARRAY E CICLI

- Posso sfruttare i cicli per scorrere l'array per modificare i valori, sfruttando una variabile contatore
- Utilizzando un ciclo `for`:

```
for (int i = 0; i < 1000; i++){
  values[i] = random(0,10);
}
```

- Oppure un ciclo `while`:

```
int n = 0;
while(n < 1000){
  values[n] = random(0,10);
  n = n + 1;
}
```

- La proprietà `length` può essere usata anche in Processing

ESERCIZIO: CREAZIONE DI UNA SCIA DEL MOUSE

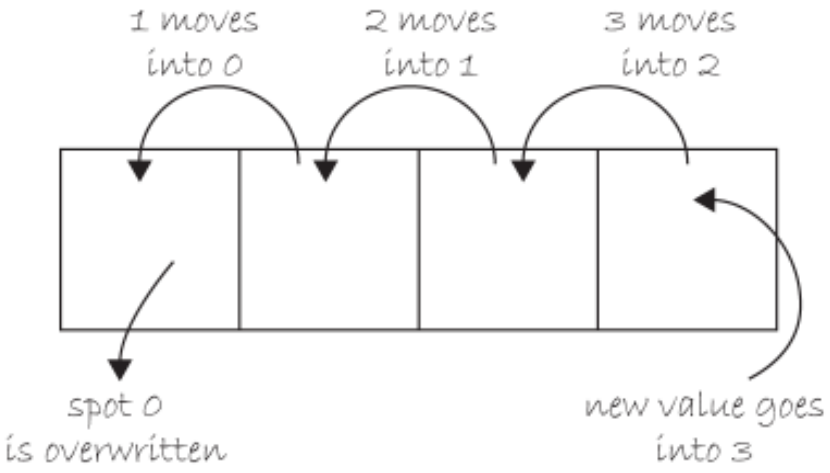
- Si vuole realizzare una striscia che si aggiorna al passaggio del mouse per simulare una scia del suo movimento
- E' necessario mantenere uno storico delle ultime posizioni per ricostruire una linea dalla posizione attuale a quella più vecchia
- Le posizione si gestiscono con due array, uno per le posizioni x e una per le posizioni y, entrambi inizializzati a zero

```
int[] xpos = new int[50];
int[] ypos = new int[50];
```

```
for (int i = 0; i < xpos.length; i++) {
  xpos[i] = 0;
  ypos[i] = 0;
}
```

ESERCIZIO: CREAZIONE DI UNA SCIA DEL MOUSE

- Ad ogni ciclo del metodo draw() bisogna aggiornare i due array delle posizioni salvando la posizione del mouse (mouseX e mouseY)
- La posizione corrente viene memorizzata nella ultima posizione dell'array
- Prima di scrivere i nuovi valori vanno spostati i valori precedenti verso sinistra (shift)



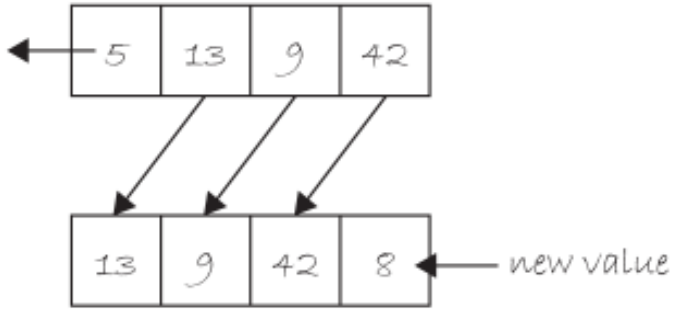


Figure 9-5

ESERCIZIO 9-8: CREAZIONE DI UNA SCIA DEL MOUSE

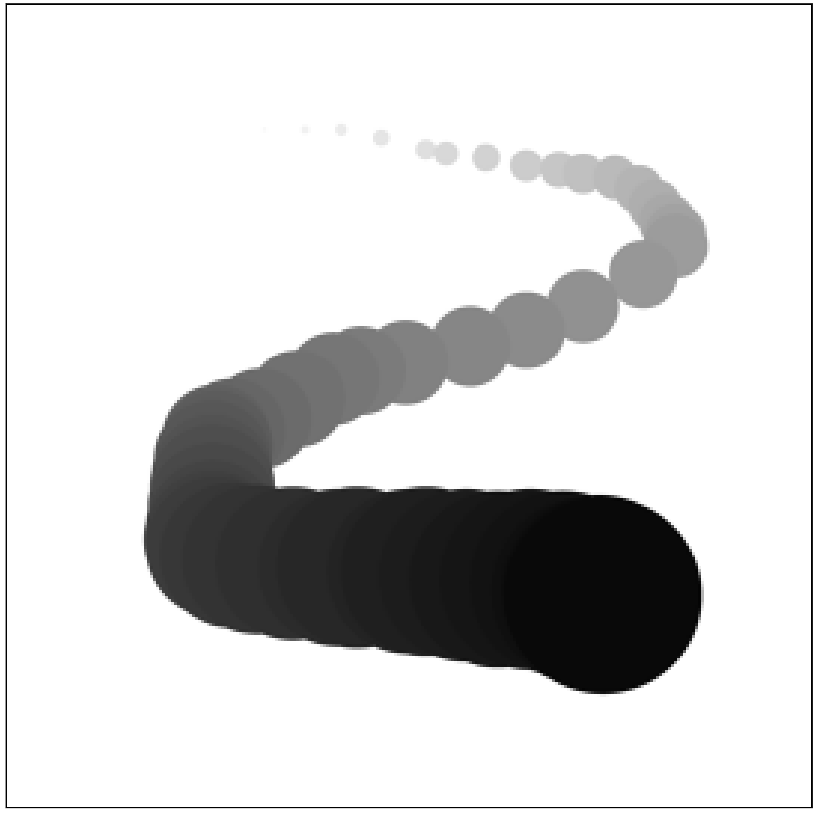


Figure 9-6

ARRAY DI OGGETTI

Before	After
`// Declare the car` `Car myCar;`	`// Declare the car array` `Car[] cars = new Car[100];`

`// Initialize the car`
`myCar = new Car(color(255), 0, 100, 2);`
	`// Initialize each element of the array`
	`for (int i = 0; i < cars.length; i++) {`
	`  cars[i] = new Car(color(i*2), 0, i*2, i);`
	`}`

`// Run the car by calling methods`
`myCar.move();`
`myCar.display();`
	`// Run each element of the array`
	`for (int i = 0; i < cars.length; i++) {`
	`  cars[i].move();`
	`  cars[i].display();`
	`}`

ARRAY DI OGGETTI - ESEMPIO

```
Car[] cars = new Car[3];

void setup(){
  size(400,400);
  for(int i=0; i<cars.length;i++){
    int c = color(random(150,255));
    int h = height/(cars.length+1)*(i+1);
    int s = (int) random(1,5);
    Car car = new Car(c,h,s);
    cars[i] = car;
  }
}

void draw(){
  background(0);
  for(int i=0; i<cars.length;i++){
    cars[i].display();
    cars[i].drive();
  }
}
```

```
class Car{
  color c;
  float xpos;
  float ypos;
  float xspeed;

  Car(int colore, int altezza, int velocita){
    c=colore;
    xpos=width/2;
    ypos=altezza;
    xspeed=velocita;
  }

  void display(){
    rectMode(CENTER);
    fill(c);
    rect(xpos,ypos,20,10);
  }

  void drive(){
    xpos=xpos+xspeed;
    if(xpos > width) xpos = 0;
  }
}
```