FUNZIONI ED EVENTI

PROGRAMMAZIONE È GESTIONE DI EVENTI

GESTIONE DI EVENTI IN ACTIONSCRIPT

```
//uso di addEventListener
oggetto.addEventListener(nomeEvento:String,
  nomeFunzione:Function);

/* esempio */
pulsante.addEventListener(“click” ,
  calcolaQuadrato);
```

PROGETTAZIONE DI UNA FUNZIONE

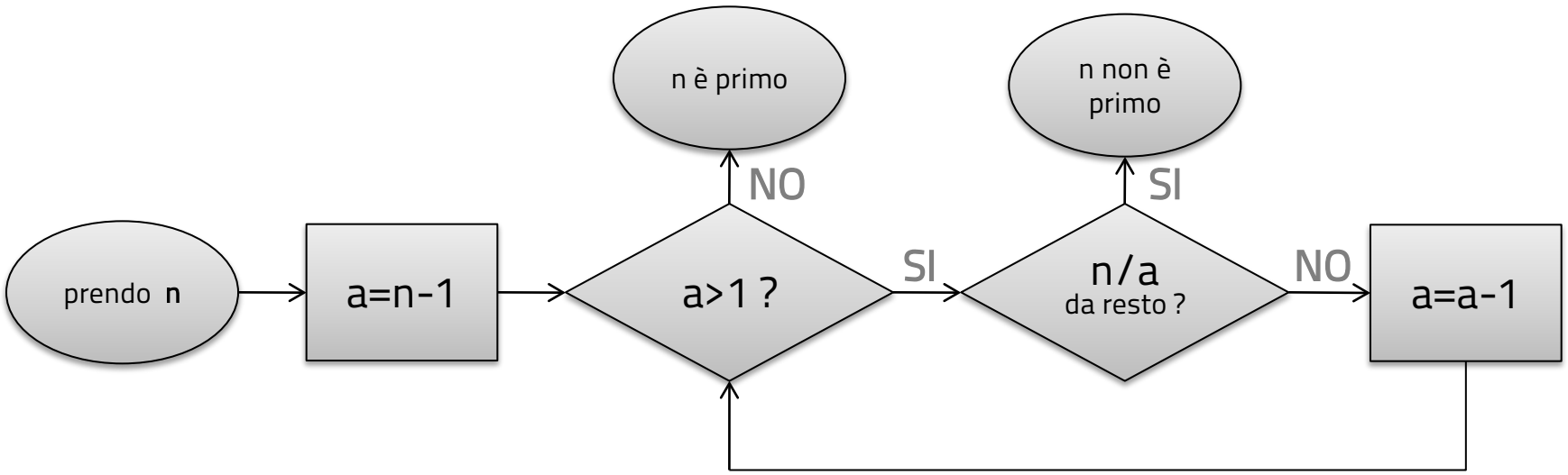
DEFINIZIONE

- Un numero primo è un numero intero che non divisori maggiori di **1** escluso se stesso.
- Per verificare se un numero che chiamerò **a** è primo dovrò controllare se esistono numeri **n** maggiori di uno e minori di **a** tali che la divisione a/n dia resto 0.
- Se non esistono tali numeri il numero è primo.

ALGORITMO

1. prendo **n** è il numero da verificare
2. definisco un numero **a** uguale ad **n -1**
3. se **a** è non è maggiore di **1** vuol dire che non esiste un divisore di **n** maggiore di **1** -> **n** è primo
4. verifico se **n/a** da resto **0**
5. se sì ho trovato un divisore di **n**: il numero non è primo
6. diminuisco **a** di **1**
7. riparto dal punto 3

NUMERO PRIMO



LA FUNZIONE PRIMO IN ACTIONSCRIPT

```
//dichiaro una funzione che accetta come parametro
//un numero e restituisce un valore Boolean (vero o falso)
function primo(n:Number):Boolean {
    /*dichiaro la variabile a di tipo number e le assegno come valore iniziale il valore di
    n (il numero che devo verificare) diminuito di 1*/
    var a:Number = n - 1;
    //inizia un ciclo che continuerà fino a che a è maggiore di 1
    while (a > 1) {
          // se a è un divisore esatto di n (cioè il resto della divisione n/a è 0)...
          if ((n % a) == 0) {
                    // n NON è un numero primo per cui restituisco il valore false
                    // la funzione termina qui e non viene eseguito alcun altro comando
                    return false;
          }
          //altrimenti (se cioè la funzione non si interrompe per effetto
          //dell'istruzione return) diminuisco a di 1 e il ciclo viene ripetuto
          a--;
    }
    // se il ciclo si concluso (a è diventato 1 per diminuzioni successive)
    // significa che non esistono divisori di n e quindi n è primo
    return true;
}
```

LE CLASSI

PROGRAMMAZIONE OOP

- Rivoluzione copernicana
  - L'approccio procedurale definisce i passi necessari per risolvere un problema
  - L'approccio ad oggetti definisce gli strumenti (oggetti) di cui ho bisogno per risolvere un problema e ne definisce le proprietà e i comportamenti.

LE PAROLE CHIAVE DELLA PROGRAMMAZIONE OOP

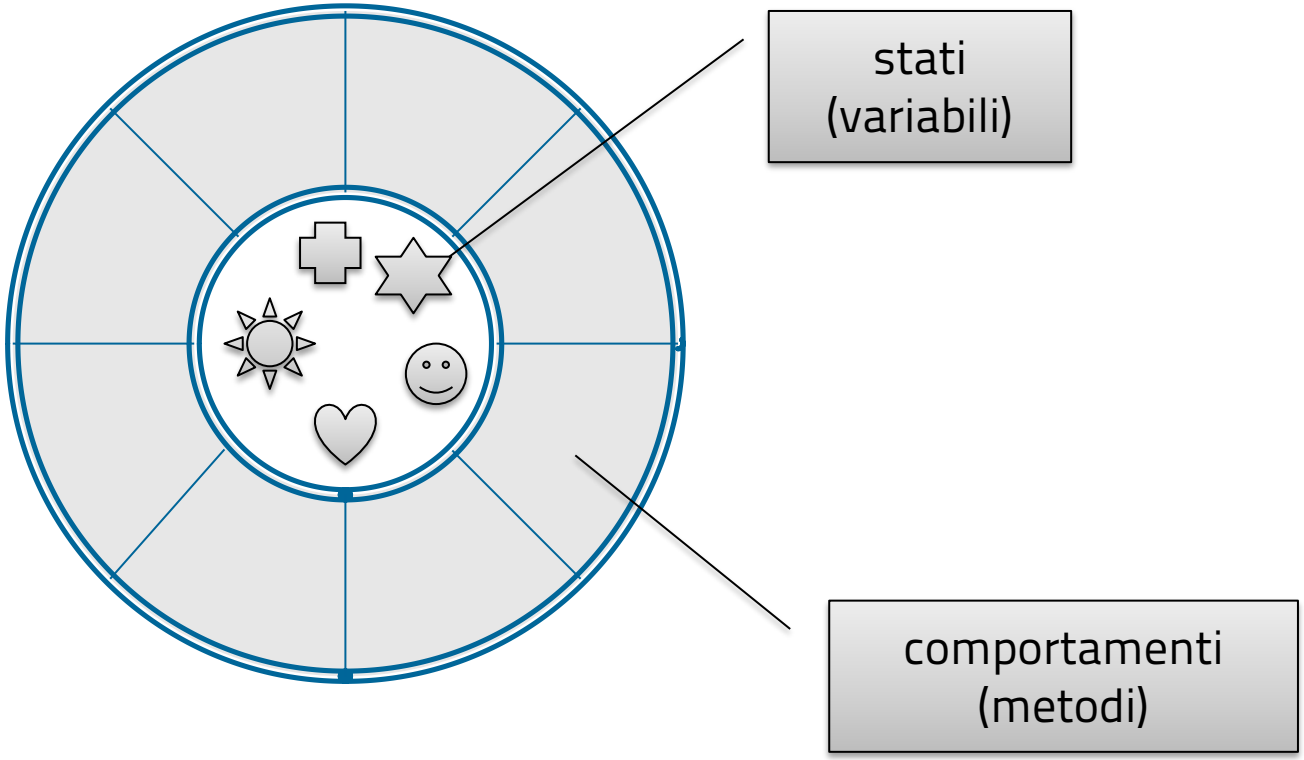
OGGETTI NEL MONDO REALE

- Un oggetto del mondo reale ha stati e comportamenti.
- Ad esempio gli stati di un gatto possono essere: nome, età e colore, i comportamenti: dormire, mangiare e fare le fusa.
- Gli stati di una bicicletta possono essere: rapporto, cadenza della pedalata, velocità e i comportamenti: cambiare rapporto, cambiare cadenza, frenare).
- Alcuni oggetti sono molto semplici (una lampada) altri più complessi (una radio).
- Alcuni oggetti possono contenere altri oggetti.

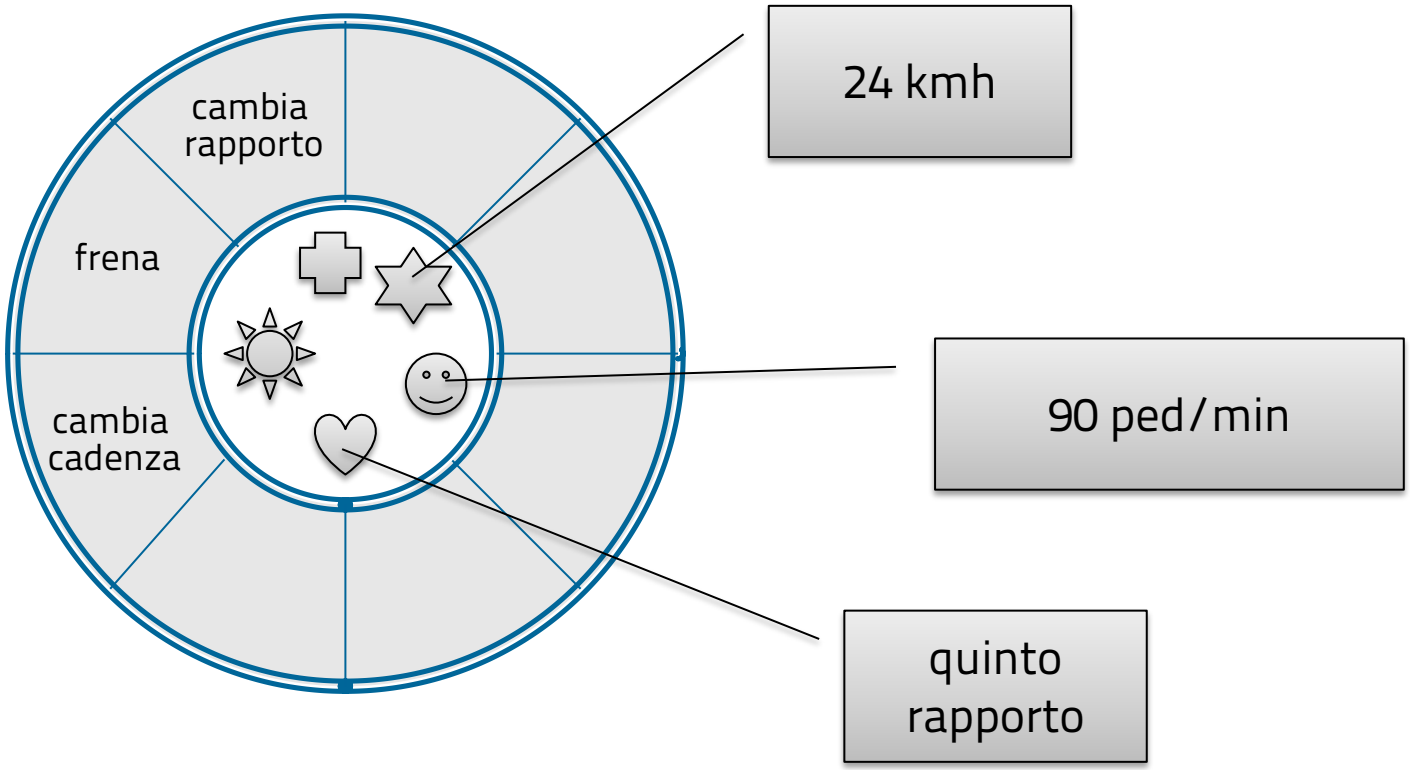
OGGETTO SOFTWARE

- Gli oggetti software sono concettualmente simili agli oggetti del mondo reale: anche loro consistono di stati e di comportamenti.
- Un oggetto memorizza il suo stato in variabili ed espone il suo comportamento attraverso *i metodi* o funzioni.
- I metodi hanno il compito di modificare gli stati interni dell'oggetto e di farlo comunicare con il mondo (software).
- Questo approccio per cui gli stati interni di un'oggeto sono nascosti e sono modificabi so usando metodi si definisce *incapsulamento dei dati* un principio fondamentale della programmazione orientata agli oggetti.

OGGETTO SOFTWARE



BICICLETTA



ISTANZE E MEMBRI DI CLASSE

- Nel mondo reale, spesso troverete molti singoli oggetti tutti dello stesso tipo. Ci possono essere migliaia di biciclette esistenti, tutti della stessa marca e modello.
- Ogni bicicletta è stata costruita sualla base dello stesso progetto e quindi contiene gli stessi componenti.
- Una classe definisce un modello per un tipo di oggetto. Tutte le caratteristiche e i comportamenti che appartengono a una classe sono detti *membri* di tale classe. In particolare, le caratteristiche (nell'esempio del gatto, il nome, l'età e il colore) sono dette *proprietà* della classe e sono rappresentate da variabili, mentre i comportamenti (mangiare, dormire) sono detti *metodi* della classe e sono rappresentati da funzioni.

EREDITARIETÀ

- Mountain bike, biciclette da strada e tandem, condividono alcune caratteristiche (velocità, cadenza della pedalata, rapporto).
- Ma ciascuno definisce anche elementi aggiuntivi che le rendono differenti:
  - i tandem hanno due posti e due set di manubri,
  - le bici da strada hanno manubrio da corsa;
  - le mountain-bikes hanno una corona aggiuntiva che consente loro rapporti di trasmissione inferiori.
- La programmazione orientata agli oggetti consente di definire classi che *ereditano* gli stati e i comportamenti di altre classi. In questo esempio, bicicletta può essere una *superclasse* e MountainBike, Bici da strada, e Tandem delle sottoclassi.

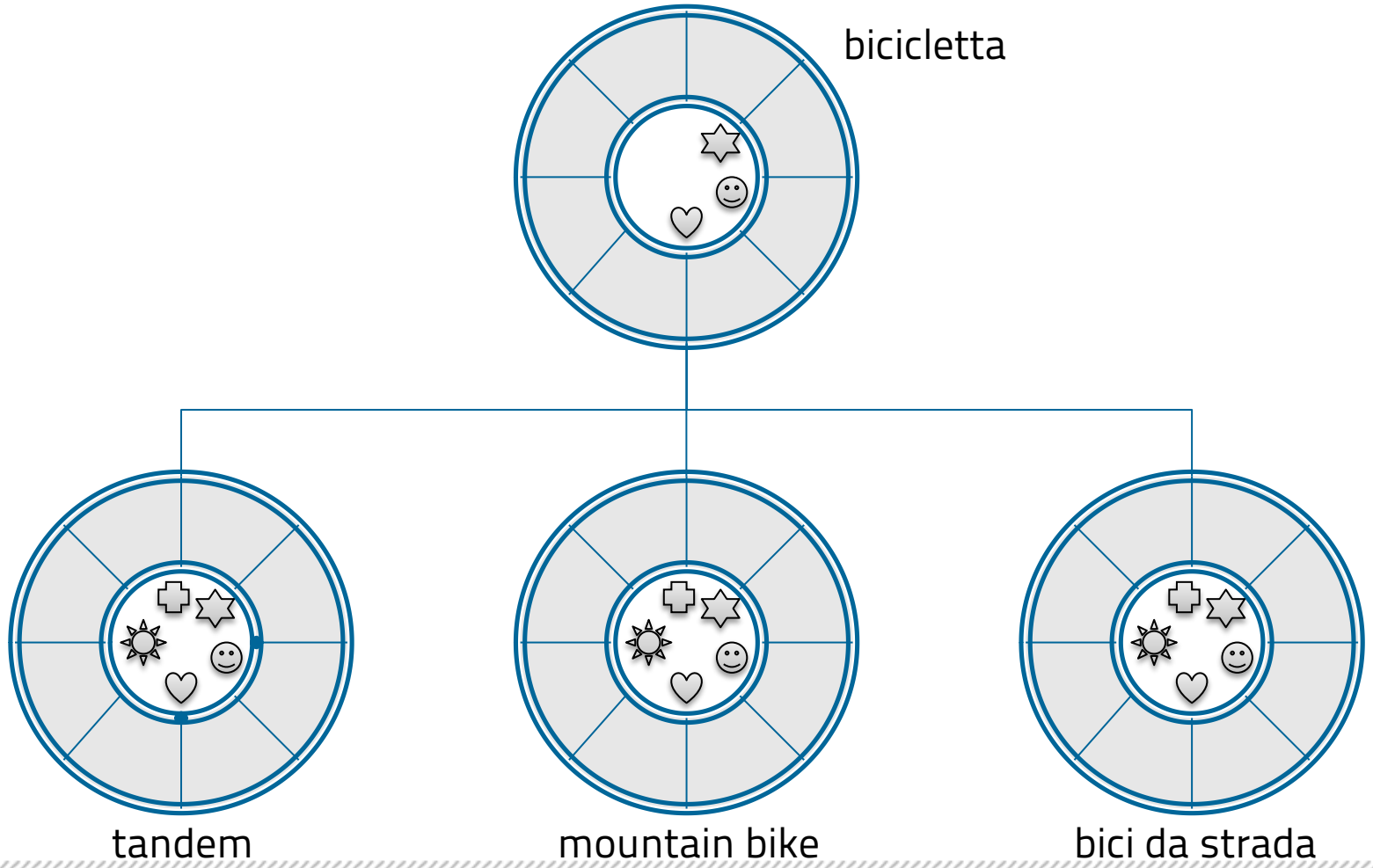
EREDITARIETÀ

- Mountain bike, biciclette da strada e bici tandem, condividono alcune caratteristiche (velocità, cadenza della pedalata, rapporto). Ma ciascuno definisce anche elemeti aggiuntivi che le rendono differenti: le biciclette tandem hanno due posti e due set di manubri, le bici da strada hanno manubrio da corsa; le mountain-bikes hanno una corona in più che consente loro rapporti di trasmissione inferiori.
- La programmazione orientata agli oggetti consente di definire classi che *ereditano* gli stati e i comportamenti di altre classi. In questo esempio, bicicletta può essere una *superclasse* e MountainBike, bici da corsa, e Tandem delle sottoclassi.
- Uno dei vantaggi principali della programmazione orientata agli oggetti è rappresentato dalla creazione di sottoclassi (tramite estensione di una classe); una sottoclasse eredita tutte le proprietà e i metodi della rispettiva classe. La sottoclasse definisce generalmente ulteriori metodi e proprietà o sostituisce metodi o proprietà definiti

EREDITARIETÀ

- Uno dei vantaggi principali della programmazione orientata agli oggetti è rappresentato dalla creazione di **sottoclassi** (tramite estensione di una classe); una **sottoclasse** eredita tutte le proprietà e i metodi della rispettiva classe.
- La sottoclasse definisce generalmente ulteriori metodi e proprietà o sostituisce metodi o proprietà definiti nella superclasse.
- Le sottoclassi possono inoltre sostituire i metodi definiti in una superclasse, ovvero fornire definizioni proprie per tali metodi.

EREDITARIETÀ



INTERFACCE

- Gli oggetti definiscono la loro interazione con il mondo esterno attraverso i metodi che espongono.
- I metodi costituiscono l'interfaccia con il mondo esterno, nello stesso modo in cui i pulsanti sulla parte anteriore del televisore, per esempio, sono l'interfaccia tra l'utente e l'elettronica che contiene involucro di plastica.
- Nella nostra classe bicicletta, l'interfaccia potrebbe essere per esempio l'elenco dei metodi che devo definire obbligatoriamente per creare una sottoclasse di bicicletta.

PACKAGE

- Un pacchetto è un namespace che organizza una serie di classi e interfacce correlate. Si può pensare di pacchetti come a diverse cartelle del computer. Poiché il software scritto in linguaggio ActionScript può essere composto da centinaia di singole classi è importante mantenere le classi organizzate in pacchetti.
- ActionScript fornisce una biblioteca (un insieme di pacchetti) che contengono le classi con cui abbiamo bisogna per gestire gli elementi. Questa libreria è noto come "Application Programming Interface", o "API".

CLASSI E TIPI

- Quando abbiamo parlato dei tipi di dati abbiamo visti che esistono *tipi di dati primitivi* e *tipi di dati derivati o complessi*.
- In un linguaggio orientato agli oggetto tutti i tipi di dati sono classi. Definiscono cioè non solo come un certo tipo di dato viene memorizzato nel computer, ma anche i metodi con cui posso interagire con esso.
- Creando una nuova classe il programmatore crea un nuovo *tipo di dati complesso*, un *modello*, cioè, di come posso gestire un determinato insieme di informazioni e di come posso interagire con esso.