

# Java

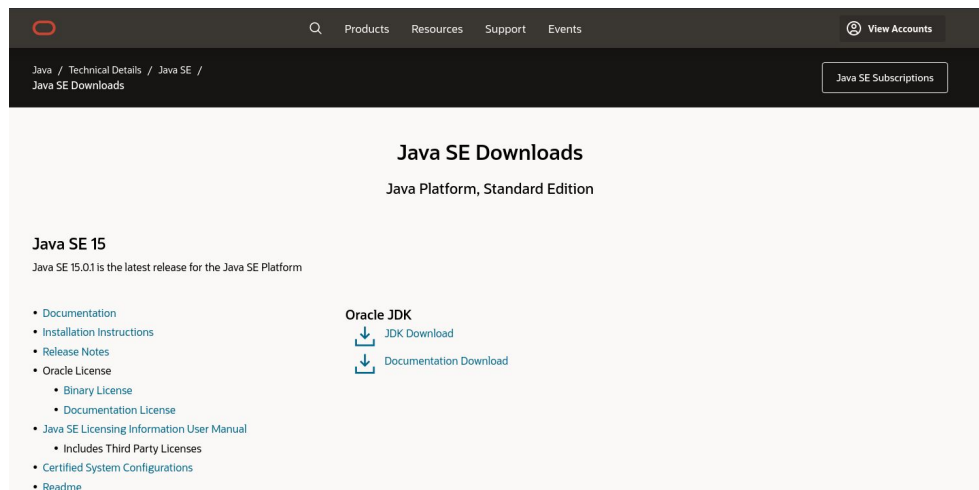
I fondamenti

# Strumenti

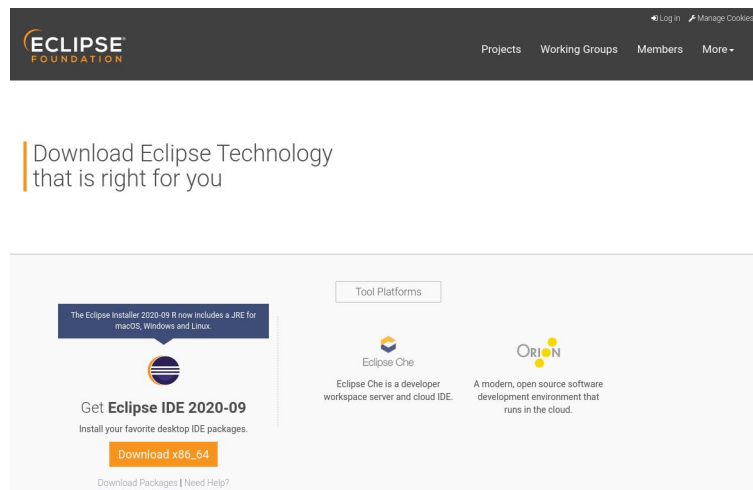


# Cosa installare per iniziare il corso

- Java JDK -> [www.oracle.com/java/technologies/javase-downloads.html](https://www.oracle.com/java/technologies/javase-downloads.html)
- Eclipse -> [www.eclipse.org/downloads/](https://www.eclipse.org/downloads/)



The screenshot shows the 'Java SE Downloads' page from Oracle. The header includes navigation links like 'Products', 'Resources', 'Support', and 'Events'. The main content area is titled 'Java SE Downloads' and 'Java Platform, Standard Edition'. It highlights 'Java SE 15' as the latest release. On the left, there is a list of links: Documentation, Installation Instructions, Release Notes, Oracle License (with sub-links for Binary License and Documentation License), Java SE Licensing Information User Manual (with a sub-link for Includes Third Party Licenses), Certified System Configurations, and Readme. On the right, under 'Oracle JDK', there are links for 'JDK Download' and 'Documentation Download'.



The screenshot shows the Eclipse Foundation website. The header features the 'ECLIPSE FOUNDATION' logo and navigation links for 'Projects', 'Working Groups', 'Members', and 'More'. The main content area is titled 'Download Eclipse Technology that is right for you'. It features a section for 'Tool Platforms' with two options: 'Eclipse Che' (described as a developer workspace server and cloud IDE) and 'ORION' (described as a modern, open source software development environment that runs in the cloud). Below this, there is a section for 'Get Eclipse IDE 2020-09' with a 'Download x86\_64' button and links for 'Download Packages' and 'Need Help?'.

# Parte 1



- Main

- Main
- Variabili

- Main
- Variabili
- Tipi primitivi

- Main
- Variabili
- Tipi primitivi
- Limite massimo/minimo dei tipi



- Main
- Variabili
- Tipi primitivi
- Limite massimo/minimo dei tipi
- Operazioni

- Main
- Variabili
- Tipi primitivi
- Limite massimo/minimo dei tipi
- Operazioni
- String

- Main
- Variabili
- Tipi primitivi
- Limite massimo/minimo dei tipi
- Operazioni
- String
- Cast

- Main
- Variabili
- Tipi primitivi
- Limite massimo/minimo dei tipi
- Operazioni
- String
- Cast
- Condizioni

- Main
- Variabili
- Tipi primitivi
- Limite massimo/minimo dei tipi
- Operazioni
- String
- Cast
- Condizioni
- Condizioni con elementi non di tipo booleano

- Main
- Variabili
- Tipi primitivi
- Limite massimo/minimo dei tipi
- Operazioni
- String
- Cast
- Condizioni
- Condizioni con elementi non di tipo booleano
- Import e User I/O

- Main
- Variabili
- Tipi primitivi
- Limite massimo/minimo dei tipi
- Operazioni
- String
- Cast
- Condizioni
- Condizioni con elementi non di tipo booleano
- Import e User I/O
- Math

# Esercizio 1.1





# La base

Scrivere un programma che:

1. Prende in ingresso 2 numeri
2. Ne stabilisca il minimo e il massimo usando la libreria math
3. Usando la libreria math generare un numero casuale compreso tra 0 e 10
4. Assegnare il numero generato precedentemente a un intero usando un cast facendo attenzione all'arrotondamento
5. Stampare un messaggio se il valore è maggiore di 5 o minore di 5
6. Calcolare il modulo di entrambi i numeri inseriti precedentemente
7. Se **almeno** uno dei due moduli calcolati è diverso da zero allora stampare il quadrato, usando la libreria math, del numero generato casualmente

# Parte 2.1



- Operatore ternario

- Operatore ternario
- Switch

- Operatore ternario
- Switch
- While/For

- Operatore ternario
- Switch
- While/For
- Methods, Modificatori, Passaggio di parametri

# Esercizio 2.1.1



# Indovina il numero

Scrivere un programma che:

1. Generi un numero casuale compreso tra 0 e 50
2. Chieda all'utente di indovinare inserendo un numero compreso da 0 a 50:
  - a. Se ha sbagliato comunicare all'utente se il numero che ha inserito è minore o maggiore e tornare al punto 2
  - b. Se ha azzeccato comunicarlo all'utente e andare al punto 3
3. Concludere il programma



# Esercizio 2.1.2



# Indovina il numero 2

Estensione 1 del precedente esercizio 2.1.1:

1. Calcoli il numero di tentativi fatti dall'utente
2. Una volta che l'utente ha indovinato comunicare il numero di tentativi fatti

Estensione 2:

1. Dopo 10 tentativi concludere il gioco e comunicare la sconfitta del giocatore

Estensione 3:

1. Chiedere se l'utente ha intenzione di giocare di nuovo
2. Contare e comunicare il numero di volte che ha giocato, il numero di volte che ha vinto e il numero di volte che ha perso

# Parte 2.2



- Operatore ternario
- Switch
- While/For
- Methods, Modificatori, Passaggio di parametri
- Array = Vettori, Matrici

- Operatore ternario
- Switch
- While/For
- Methods, Modificatori, Passaggio di parametri
- Array = Vettori, Matrici
- Try/Catch, Errori

- Operatore ternario
- Switch
- While/For
- Methods, Modificatori, Passaggio di parametri
- Array = Vettori, Matrici
- Try/Catch, Errori
- Classi, metodi, accesso

# Esercizio 2.2.1



# Gestione Persone

Scrivere un programma che:

1. Creare una classe Persone (nome, eta')
2. Creare un array di 5 persone
3. Stampare l'array
4. Ordinare l'array in base all'eta' in modo crescente (dal più piccolo al più grande)
5. Chiedere in input un nuovo nome per ogni persona
6. Eseguire dei controlli sugli input inseriti usando il try catch
7. Stampare la lista di persone



# Esercizio 2.2.2



# Gestione Persone 2

REFACTORING codice esercizio 2.2.1:

1. estrarre ogni forma di duplicazione dal codice in metodi
2. è possibile creare una sorta di classe di utility per le persone?

# Esercizi di fine sezione



# Esercizi di fine capitolo 2

1. Inverti una stringa
2. Inverti una frase ("pizza con il prezzemolo" -> "prezzemolo il con pizza")
3. Trova il minimo e il massimo in un vettore
4. Trova il minimo e il massimo in una matrice
5. A partire da un vettore restituisci i suoi elementi senza duplicati ("1 3 5 3 7 3 1 1 5" -> "1 3 5 7")
6. A partire da un vettore restituisci i singoli elementi con il conteggio delle volte che sono presenti nel vettore (il vettore del punto sopra diventa "1(3) 3(3) 5(2) 7(1)")
7. Fizz Buzz -> [https://en.wikipedia.org/wiki/Fizz\\_buzz](https://en.wikipedia.org/wiki/Fizz_buzz)

# Parte 3



- LinkedList

- LinkedList
- Ereditarietà, Polimorfismo, Modificatori

- LinkedList
- Ereditarietà, Polimorfismo, Modificatori
- Interfacce



- LinkedList
- Ereditarietà, Polimorfismo, Modificatori
- Interfacce
- Classi astratte

- LinkedList
- Ereditarietà, Polimorfismo, Modificatori
- Interfacce
- Classi astratte
- Enumerazioni

- LinkedList
- Ereditarietà, Polimorfismo, Modificatori
- Interfacce
- Classi astratte
- Enumerazioni
- Modificatore static

# Esercizi di fine sezione



# Progetto di fine capitolo

Scegliere uno di questi progetti e usare le conoscenze che avete appreso:

1. Creare un programma per gestire una lista della spesa
2. Creare un gestionale dei dipendenti di una azienda
3. Calcolatrice completa prendendo in input la stringa da calcolare (considerare anche calcoli trigonometrici, integrali, etc.etc.)
4. TODO list
5. Ricettario vuoto in cui posso inserire nuove ricette, permettere al ricettario di ricordarsi le ricette nuove inserite usando file di sistema inseriti in una cartella del filesystem
6. Riprodurre il gioco dama
7. Riprodurre il gioco forza 4

# Parte 4




# Modelli - Design pattern

Si tratta di metodi di risoluzione a problemi di ingegneria del software comuni (ideati inizialmente dalla GoF).

Trattano risoluzioni efficaci per la progettazione del software != algoritmi risolutivi a tutti i problemi.

Sono classificati in 3 categorie principali:

- Creazionali - Creational
- Strutturali - Structural
- Comportamentali - Behavioral

- 
- nome
  - problema che risolvono
  - soluzione al problema
  - conseguenze delle soluzione

# Creazionali

Problema = fornire la capacità di creare oggetti sulla base di un criterio richiesto e in modo controllato.

Soluzioni:

- **Abstract factory**
- Builder
- **Dependency injection**
- **Factory method**
- Lazy initialization
- Object pool
- Prototype
- Resource Acquisition Initialization
- **Singleton**
- Multiton



# Strutturali

Problema = organizzare classi e oggetti diversi per formare strutture più grandi e fornire nuove funzionalità.

Soluzioni:

- **Adapter**
- **Bridge**
- **Composite**
- **Decorator**
- Extension object
- Proxy
- Facade
- Flyweight
- Front Controller
- Marker
- Module
- **Twin**

# Comportamentali

Problema = identificazione di modelli di comunicazione comuni tra gli oggetti e la loro realizzazione.

Soluzioni:

- **Strategy**
- Blackboard
- Chain of Responsibility
- **Command**
- Interpreter
- **Iterator**
- **Mediator**
- State
- **Template method**
- **Visitor**
- **Memento**
- Null Object
- **Observer (publish/subscribe)**
- Servant
- Specification

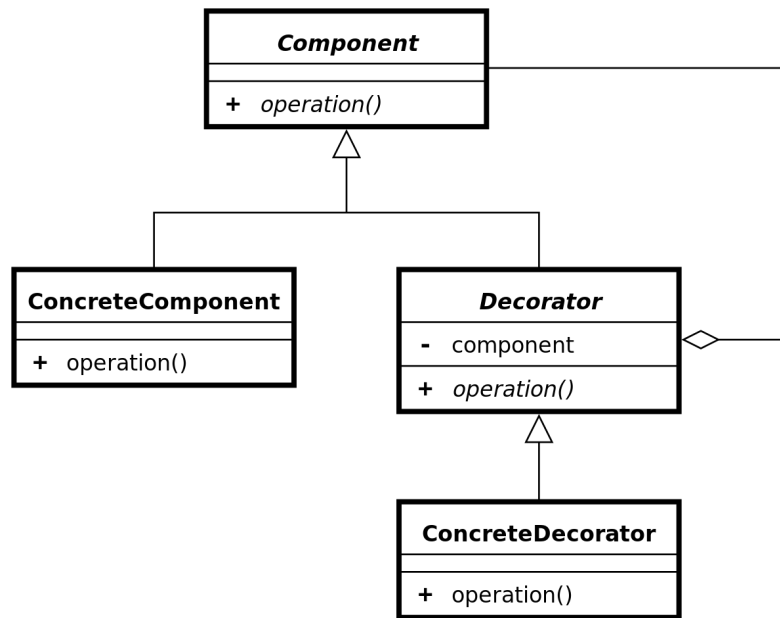
# Esempio 1



# Decorator

Problema che risolve: Aggiungere dinamicamente ulteriori responsabilità ad un oggetto.

Soluzione: Definire una classe decoratore che aggiunge nuove funzionalità al componente principale.



# Esempio: Pizza

Assumiamo che una pizza base non abbia nessun condimento.

Abbiamo diversi tipi di pizza di cui conosciamo i suoi condimenti: margherita, napoli, quattro formaggi, etc.etc.

Se volessimo aggiungere altri condimenti?

⇒ Decorator

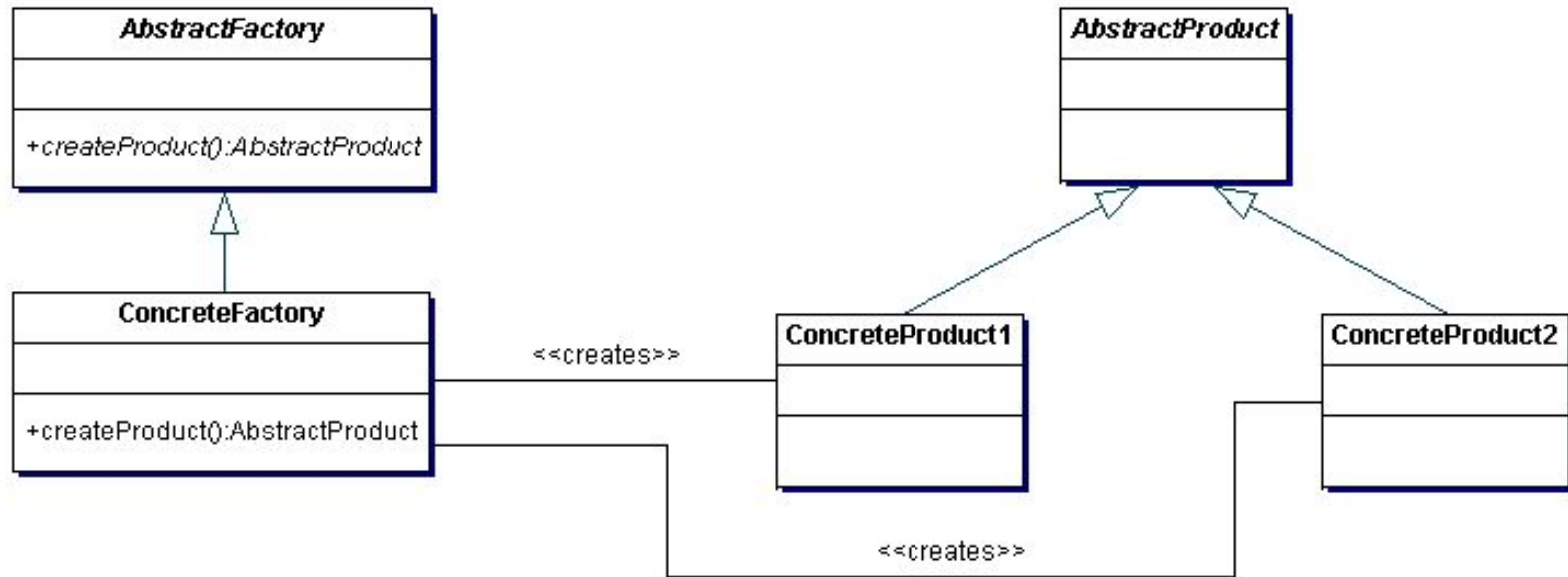
# Esempio 2



# Abstract Factory

Problema che risolve: fornire un'interfaccia per la creazione di famiglie di oggetti correlati o dipendenti senza specificare le loro classi concrete.

Soluzione: incapsulare un gruppo di elementi che sono in comune tra di loro senza specificare le loro classi concrete.



# Esempio: Kingdom Factory

Un regno, di qualsiasi tipo sia, è composto da: un castello, un re e un esercito.

⇒ abstract factory

Creiamo un factory per il regno elfico.

Creiamo un factory per il regno degli orchi.

Creiamo un factory per i 2 factory sopra.

Testiamo il comportamento del factory.



FINE

