Mauro Bogliaccino

Corso Java - Unità formative

Programma del corso Java SE - Java SE 11 Programmer II

Java SE 11 Programmer II

Finanziato dall'Unione europea NextGenerationEU

Introduzione al corso di programmazione orientato agli oggetti in linguaggio Java.

- Corso Java
  - Che cos'è un programma Java
  - Comprensione della tecnologia e dell'ambiente Java
  - Creare una classe main Java
  - Input/Output
  - Lavorare con i dati
  - Identificatori e tipi di dato

- Le variabili in java
- Tipi di dato primitivi
- Gestire più elementi
- Introduzione alla libreria standard
- Strutture condizionali
- Utilizzo di operatori e costrutti decisionali
- Lavorare con gli Array in Java
- Lavorare con tipi di dati primitivi Java e API String
- Tipi complessi

- Everything is Object
- Manipolare e formattare i dati nel programma
- La classe Math
- Descrivere oggetti e classi
- Creare e utilizzare i metodi
- I metodi costruttori
- Utilizzare l'incapsulamento
- La Programmazione ad oggetti
- Classi istanziabili

- Associazione o uso
- Aggregazione
- Composizione
- Estensione
- Ereditarietà in Java
- Polimorfismo
- Utilizzare le interfacce
- Programmazione astratta attraverso le interfacce
- Gestione delle eccezioni e asserzioni
- Gestire le eccezioni II

- testing e debugging
- JShell
- Programmazione Funzionale
- Stream e stream paralleli
- Elementi Sintassi JAVA
- Advanced Class design
- Collezioni e generici
- design patterns
- JAVA WEB

- Fondamenti di Java
- Interfacce Java
- Generici e Collezioni
- Framework Collections
- Interfaccia funzionale ed espressioni lambda
- API Java Stream
- Interfacce funzionali integrate
- Operazioni Lambda in streaming

- Programmazione API Java e concetti di codifica sicura
- I/O (Fundamentals e NIO2)
- Input/Output Stream
- Codifica sicura nell'applicazione Java SE
- Applicazioni di database con JDBC
- Elementi principali JDBC
- Localizzazione
- annotazioni
- JSP e Servlet
- Distribuire e mantenere un'applicazione

- Compredere l'uso dei moduli
- Programmazione modulare
- Comprensione dei moduli
- Migrazione ad un'applicazione modulare
- Servizi in un'applicazione modulare
- Concorrenza
- Flusso parallelo
- Spring Boot

Che cos'è un programma Java

- Caratteristiche principali del linguaggio Java
- Tecnologia Java e ambiente di sviluppo
- Esecuzione e test di un programma Java

Comprensione della tecnologia e dell'ambiente Java

- Descrivere la tecnologia Java e l'ambiente di sviluppo Java
- Identificare le funzionalità chiave del linguaggio Java
- struttura del **JDK**
- Java: le basi del linguaggio
- Introduzione linguaggio Java
- Introduzione a Java e al processo di compilazione
- Compilazione ed esecuzione

Creare una classe main Java

- Classi Java
- Il metodo main
- Aggiunta di un metodo main
- Crea un programma Java eseguibile con una classe main
- Compilare ed eseguire un programma Java dalla riga di comando
- Creare e importare pacchetti
- Primo approccio al codice
- metodo main e metodi ausiliari
- Primi programmi Java.

Input/Output

- Input/Output utente
- Input da console, java.util.Scanner
- Output con oggetto System.out

Lavorare con i dati

- Presentazione delle variabili
- Lavorare con le stringhe
- Lavorare con i numeri
- Manipolazione dei dati numerici

Identificatori e tipi di dato

- Schema Libero
- Case sensitive
- Commenti
- Regole per gli identificatori

Le variabili in java

- Dichiarazione di una variabile:
- Variabili d'istanza
- Variabili locali
- Scope delle variabili
- Parametri formali
- Argomenti passati al metodo main

Tipi di dato primitivi

- Tipi di dati interi, casting e promotion
- Tipi di dati a virgola mobile, casting e promotion
- Tipo di dato logico - booleano
- Tipo di dato primitivo letterale

Gestire più elementi

- Lavorare con le condizioni
- Utilizzo delle istruzioni IF
- Lavorare con un elenco di elementi
- Elaborazione di un elenco di elementi

Introduzione alla libreria standard

- Il comando import
- La classe String
- La documentazione della libreria standard di Java
- Cicli finiti, infiniti, annidati

Strutture condizionali

- Operatori relazionali e condizionali
- Altri modi per usare i costrutti IF / ELSE
- Utilizzo delle istruzioni switch
- Utilizzo del debugger

Utilizzo di operatori e costrutti decisionali

- Utilizzare gli operatori Java: uso delle parentesi per sovrascrivere la precedenza dell'operatore
- Utilizzare le istruzioni di controllo Java: if, else e switch
- Crea e usa
  - do...while,
  - while,
  - for
  - foreach,
  - loop nidificati,
  - istruzioni break e continue

Lavorare con gli Array in Java

- Gli array in Java
  - Dichiarazione
  - Creazione
  - Inizializzazione
- Array **monodimensionali**
- Array **bidimensionali**
- Limiti degli array in JAVA: **dimensione prefissata**
- **Copia** di array
- Array e ArrayList.

Lavorare con tipi di dati primitivi Java e API String

- Dichiarare e inizializzare le variabili (casting e promotion di tipi primitivi)
- Identificare l'ambito della variabile
- Usa l'inferenza del tipo di variabile locale
- Crea e manipola stringhe
- Manipola i dati usando la classe StringBuilder e i suoi metodi
- Caso particolare: la classe String e i principali metodi.

Tipi complessi

- struttura delle classi,
- progettazione delle classi in UML,
- stato interno: proprietà, attributi, fields
- metodi: le azioni che gli oggetti compiono o che possiamo compiere inviando un messaggio

Everything is Object

- Classe Object
  - Metodi equals, toString e clone.
- Classi wrapper per trasformare in oggetti i tipi di dati primitivi
- La classe Random: generazione di numeri pseudo-random.
- La classe Timer: schedulare un'azione, che verrà eseguita ogni tot millisecondi.

Manipolare e formattare i dati nel programma

- Utilizzare la classe String
- Utilizzo dei documenti dell'API Java
- Uso della classe StringBuilder
- Ulteriori informazioni sui tipi di dati primitivi
- Più operatori numerici
- promotion e casting delle variabili

La classe Math

- La classe Math e i principali metodi.
- uso di una classe con metodi statici
- accedere a proprietà e metodi senza istanziare nuovi oggetti

Descrivere oggetti e classi

- Lavorare con oggetti e classi
- Definizione di campi e metodi
- Dichiarazione, istanziazione e inizializzazione di oggetti
- Lavorare con i riferimenti agli oggetti
- Lavorare con le matrici
- Dichiarare e creare istanze di oggetti Java e spiegare i cicli di vita degli oggetti (inclusa la creazione, la dereferenziazione mediante riassegnazione e la garbage collection)
- Definire la struttura di una classe Java

Creare e utilizzare i metodi

- Utilizzare metodi
- Argomenti del metodo e valori restituiti
- Come vengono passati gli argomenti a un metodo
- Metodi static
- Overload di un metodo

I metodi costruttori

- Caratteristiche di un costruttore
- Costruttore di default
- Overload di Costruttori

Utilizzare l'incapsulamento

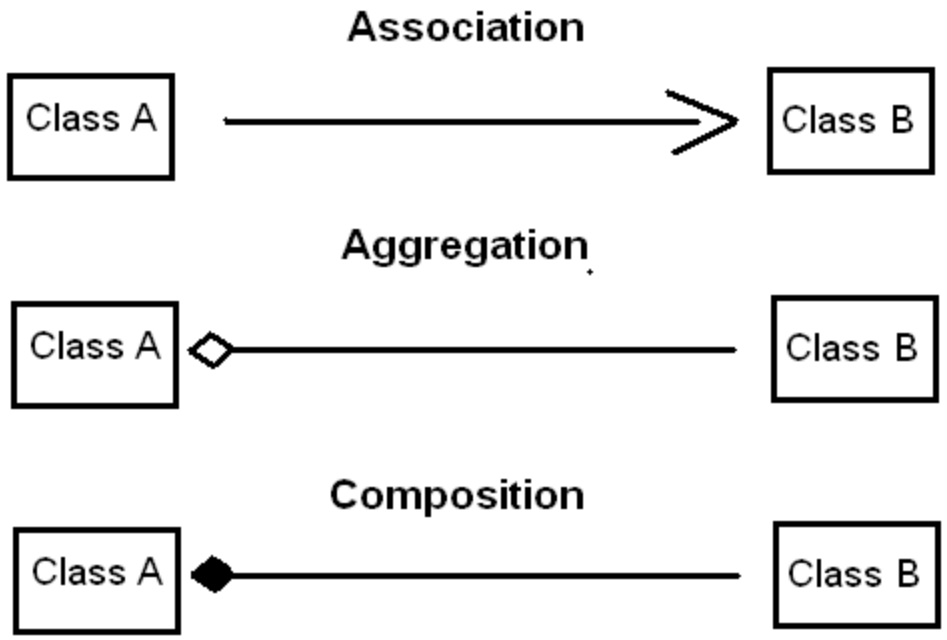
- Controllo di accesso
- Modificatori d'accesso: package, public, private e protected
- Incapsulamento
  - metodi getter e setter
- Package e visibilità

- Progettazione di **nuove** classi e metodi.
- Oggetti, variabili, riferimenti
- Classi, metodi e variabili di istanza
- Istanziare un oggetto: il metodo costruttore
- Variabili d'istanza e incapsulamento
- Visibilità dei membri di una classe
- La parola riservata null
- Coesione, dipendenza, programmazione per contratto, parametro implicito, effetti collaterali
- Membri statici e membri dinamici
- Modificatori d'accesso e visibilità: public, private, friendly, static
- Classi e metodi con parametri

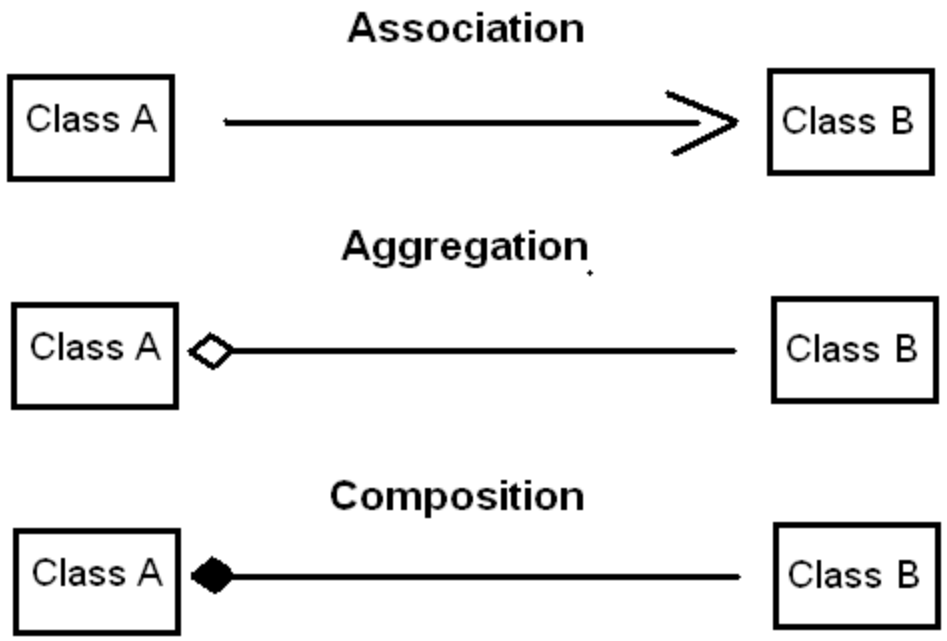
Classi istanziabili

- tipo di utilizzo classe
  - creo oggetto,
  - attraverso l'oggetto, con l'opertore dot (.) accedo ai suoi metodi e variabili
- relazioni tra classi: uso, aggregazione, teoria delle classi

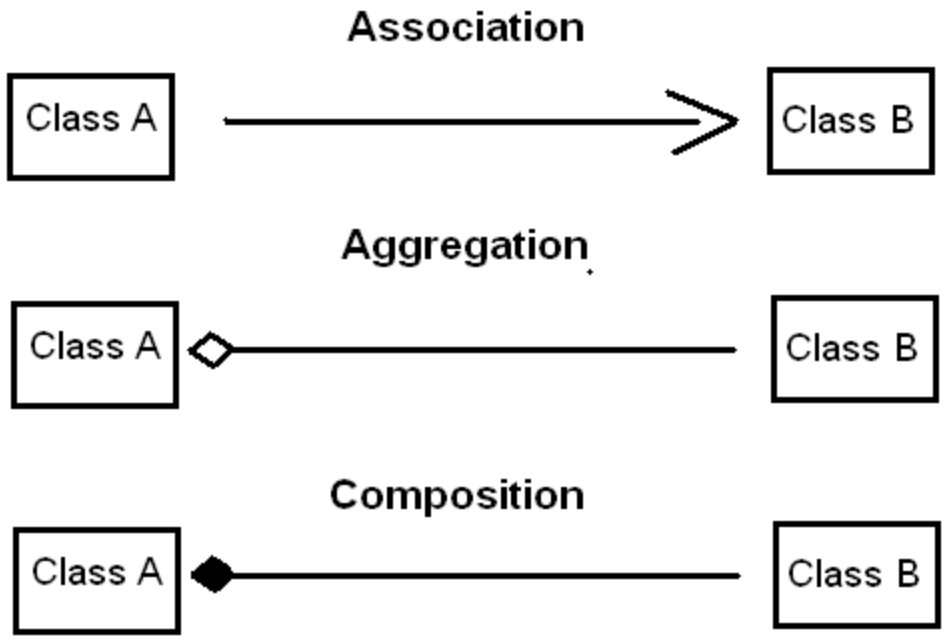
Associazione o uso



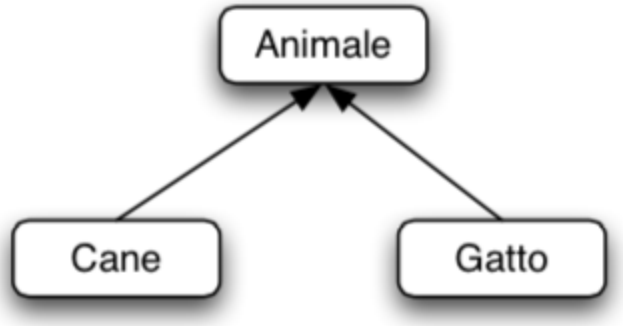
Aggregazione



Composizione



Estensione



- Teoria:
  - Class design: diversi ruoli degli oggetti/attori del progetto
  - diverse classi con differenti relazioni tra esse
  - annotazioni @override (solo in caso di extends), p.es. `toString()`

Ereditarietà in Java

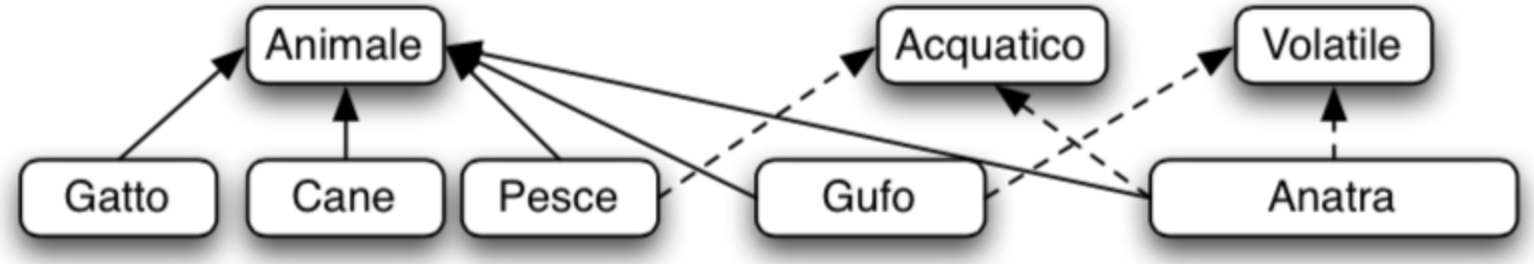
- Riutilizzo del codice.
- Strutture ereditarie
- La parola riservata super
- Impedire l'ereditarietà: la parola riservata final
- Creare e utilizzare sottoclassi e superclassi
- Creare ed estendere le classi astratte

Polimorfismo

- Utilizzare il polimorfismo per lanciare e chiamare metodi, differenziando il tipo di oggetto rispetto al tipo di riferimento
- Distinguere tra overload, override e information hiding

Utilizzare le interfacce

- Utilizzo delle interfacce
- Inferenza di tipo variabile locale
- Utilizzare l'interfaccia List
- Presentazione delle espressioni Lambda



Programmazione astratta attraverso le interfacce

- Creare e implementare interfacce
- Distinguere l'ereditarietà delle classi dall'ereditarietà dell'interfaccia comprese le classi astratte
- Dichiarare e utilizzare le istanze List e ArrayList
- Comprensione dell'espressione lambda
- interfacce

Gestione delle eccezioni e asserzioni

- Panoramica
- Gestione degli errori
- Asserzioni
- Catturare e lanciare eccezioni
- Gestione di più eccezioni ed errori
- Descrivere i vantaggi della gestione delle eccezioni e differenziare tra eccezioni verificate, non selezionate ed errori
- Propagazione di eccezioni

Gestire le eccezioni II

- Quando e come usare la gestione eccezioni di JAVA
- Crea e invoca un metodo che genera un'eccezione
- le parole chiave throws e throw
- try-with-resources `try(resources){...} catch(Exception e){...}`
- blocco try-catch-finally
- Creare blocchi try-catch e determinare in che modo le eccezioni alterano il flusso del programma

testing e debugging

- Cenni sull'uso di un debugger
- Il debugger di Eclipse
- Correzione di un programma Java.
- Test unitari
- Junit
- TestNG
- Selenium

JShell

- Codice di test
- Nozioni di base di JShell
- JShell in un IDE

Programmazione Funzionale

- Interfacce funzionali ed espressioni lambda
- Nuove interfacce funzionali integrate
- Stream di collezioni e filtri

Stream e stream paralleli

- Concorrenza
- Flussi paralleli
- Operazioni terminal: Collezionisti
- Creazione di stream personalizzati

Elementi Sintassi JAVA

- Java Advanced: enumerazioni
- Java Advanced: inizializzatore statico
- Creare Jar eseguibili, creare e includere Jar nel progetto
- Java advanced:
  - nested classes,
  - member classes,
  - anonymous inner classes
- Reflection, Class, Constructor classes

Advanced Class design

- Binding dinamico
- Casting con oggetti
- Cast e instanceof
- Classi astratte
- Classi innestate o interne
- Schede CRC e diagrammi UML.
- Classi anonime

Collezioni e generici

- Cos'è il Java Collections Framework (JCF)?
- Iteratori
- gestire le collezioni: List, Set e Map
- I metodi di utilità di Collections
- List:
  - Vector, ArrayList
  - LinkedList
- Set:
  - HashSet, TreeSet
- Queue: PriorityQueue, Deque
- Cosa sono i generici?

design patterns

- SOLID
- GOF
- singleton
- decorator
- factory
- mvc
- dao

JAVA WEB

- introduzione JSP
- approfondimento JSP
- JSP e Servlet
- [JSP: le direttive]
- [JSP: le direttive]
- [JSP: le espressioni]
- [JSP: le dichiarazioni]
- GlassFish e TomCat
- JSTL: Jsp standard action
- Servlet REST controller: doGet, doPost
- RequestDispatcher

Fondamenti di Java

- Creare e utilizzare le classi final
- Creare e utilizzare classi interne, nidificate e anonime
- Creare e utilizzare le enumerazioni

Interfacce Java

- Creare e utilizzare interfacce con metodi predefiniti (default)
- Creare e utilizzare interfacce con metodi privati

- Utilizzare le classi wrapper, il boxing automatico e l'unboxing automatico
- Creare e utilizzare classi generiche, metodi con notazione a diamante e caratteri jolly
- Descrivi il Framework di raccolta e usa le interfacce di raccolta chiavi
- Usa Comparator e interfacce comparabili
- Creare e utilizzare metodi di praticità per le raccolte

Framework Collections

- Foreach ed Iterator
- Implementazioni di Set e SortedSet
- Implementazioni di List
- Implementazioni di Queue
- Implementazioni di Map e SortedMap
- Tipi Generics

Interfaccia funzionale ed espressioni lambda

- Definire e scrivere interfacce funzionali
- Crea e usa espressioni lambda tra cui istruzione lambdas, variabile locale per i parametri lambda

API Java Stream

- Descrivere l'interfaccia Stream e le pipeline
- Usa espressioni lambda e riferimenti a metodi

Interfacce funzionali integrate

- Utilizzare le interfacce funzionali di base tra cui Predicato, Consumatore, Funzione e Fornitore
- Usa le variazioni primitive e binarie delle interfacce di base del pacchetto java.util.function

Operazioni Lambda in streaming

- Estrai i dati del flusso usando i metodi map, peek e flatMap
- Cerca i dati dello stream utilizzando i metodi di ricerca findFirst, findAny, anyMatch, allMatch e noneMatch
- Usa la classe opzionale
- Eseguire calcoli utilizzando le operazioni di conteggio, max, min, media e somma del flusso
- Ordina una raccolta usando le espressioni lambda
- Utilizzare i raccoglitori con stream, inclusi le operazioni groupingBy e partitioningBy

Programmazione API Java e concetti di codifica sicura

- I/O Input/Output (Fundamentals e NIO2)
- Codifica sicura
- Applicazioni di database con JDBC
- Localizzazione

I/O (Fundamentals e NIO2)

- Leggi i dati e scrivi i dati della console e dei file utilizzando il flusso I / O
- Utilizzare I / O Stream per leggere e scrivere file
- Leggere e scrivere oggetti utilizzando la serializzazione
- Utilizzare l'interfaccia Path per operare su percorsi di file e directory
- Utilizzare la classe Files per controllare, eliminare, copiare o spostare un file o una directory
- Usa l'API Stream con i file

Input/Output Stream

- Introduzione all'I/O: input da tastiera
- Java.io: Leggere un file
- Java.io: Scrivere su un file
- Java.io:Operazioni su file
- Networking: Socket
- Flussi di byte e di caratteri
- Flussi di oggetti
- Accesso sequenziale e random.
- Files
- Serializzazione e deserializzazione.

Codifica sicura nell'applicazione Java SE

- Prevenzione della negazione del servizio nelle applicazioni Java
- Protezione delle informazioni riservate nell'applicazione Java
- Implementazione delle linee guida per l'integrità dei dati - iniezioni e convalida dell'inclusione e dell'input
- Prevenire l'attacco esterno del codice limitando l'accessibilità e l'estensibilità, gestendo correttamente la convalida dell'input e la mutabilità
- Protezione della costruzione di oggetti sensibili
- Protezione della serializzazione e della deserializzazione

Applicazioni di database con JDBC

- Collegati ai database utilizzando gli URL JDBC e DriverManager
- Utilizzare PreparedStatement per eseguire operazioni CRUD
- Convenzioni JDBC URL Naming
- Gestione dei driver: il DriverManager
- Gestione degli errori: le SQLException
- Supporto per i tipi di dati
- Estensioni standard di JDBC
- Connection Pooling
- Crud su DB

Elementi principali JDBC

- Scaricare il connector-J
- Connection
- DriverManager
- Statement, PreparedStatement, CallableStatement
- ResulSet
    - executeQuery()
    - executeUpdate()

Localizzazione

- Usa la classe Locale
- Usa pacchetti di risorse
- Formatta messaggi, date e numeri con Java

annotazioni

- Descrivere lo scopo delle annotazioni e dei tipici schemi di utilizzo
- Applica annotazioni a classi e metodi
- Descrivi le annotazioni di uso comune nel JDK
- Dichiara annotazioni personalizzate

JSP e Servlet

- Primo approccio a JSP
- Installazione ed esecuzione della prima pagina JSP
- Elementi fondamentali di JSP
- Utilizzo degli elementi fondamentali
- Utilizzo di JavaBeans
- Lavorare con i database
- Elementi Avanzati di una pagina JSP
- Uso di Etichette personalizzate

Distribuire e mantenere un'applicazione

- Pacchetti, JAR, architettura
- Modifica e requisiti dell'applicazione

Compredere l'uso dei moduli

- Il sistema del modulo
- JARs
- Dichiarazioni del modulo
- JDK modulare

Programmazione modulare

- Introduzione alla programmazione modulare in Java
- Servizi in un'applicazione modulare
- Migrazione ad un'applicazione modulare

Comprensione dei moduli

- Descrivi il JDK modulare
- Dichiarare i moduli e abilitare l'accesso tra i moduli
- Descrivi come viene compilato ed eseguito un progetto modulare

Guida su baeldung.com

Migrazione ad un'applicazione modulare

- Migrare l'applicazione sviluppata utilizzando una versione Java precedente a SE 9 a SE 11 inclusa la migrazione top-down e bottom-up, suddividendo un'applicazione Java SE 8 in moduli per la migrazione
- Esegui un'applicazione modulaized su classpath e su modulepath
- Utilizzare jdeps per determinare le dipendenze e identificare il modo per affrontare le dipendenze cicliche

Servizi in un'applicazione modulare

- Descrivere i componenti dei servizi, comprese le direttive
- Progetta un tipo di servizio,
- carica i servizi utilizzando ServiceLoader,
- verificare le dipendenze dei servizi inclusi i moduli consumer e provider

Concorrenza

- Creare thread di lavoro utilizzando Runnable, Callable e utilizzare un ExecutorService per eseguire contemporaneamente attività
- Usa java util raccolte e classi simultanee tra cui CyclicBarrier e CopyOnWriteArrayList
- Scrivi codici thread-safe
- Identificare i problemi di threading come deadlock e livelock

Flusso parallelo

- Sviluppa il codice che utilizza il flusso parallelo
- Implementa decomposizione e riduzione con stream

Spring Boot

- Spring framework
- Spring MVC
- Spring Boot
- Dependency Injection
- Spring Data
- Hibernate
- Thymeleaf