

Curs java pentru testare automata

JAVA

Cuprins curs:

- ▶ Final keyword
- ▶ Super keyword
- ▶ Inheritance

Final keyword

- ▶ Cuvântul cheie **final** declarat pentru variabilă, metodă și clasă indică faptul că „Acest lucru nu poate fi modificat”.
- ▶ Valoarea variabilei finale nu poate fi modificată.
- ▶ O metodă finală nu poate fi suprascrisă.
- ▶ O clasă finală nu poate fi moștenită.

Final keyword

- ▶ Final este un cuvânt cheie care este utilizat pentru a restricționa utilizatorul în programarea Java.
- ▶ Este un modifier fără acces care se aplică numai unei variabile, metode sau clase.
- ▶ Cuvântul cheie Java Final are trei utilizări diferite:
 - ▶ 1. A declara o constantă.
 - ▶ 2. Pentru a preveni moștenirea.
 - ▶ 3. Pentru a preveni suprascrierea metodei.

Final keyword

Final variable

valoarea nu poate fi
modificata

Final method

nu poate fi suprascrisa

Final class

nu poate fi mostenita

Final variable

- ▶ O variabilă declarată cu un cuvânt cheie final este cunoscută ca o variabilă finală în Java.
- ▶ Variabila finală înseamnă o constantă (valoarea nu poate fi modificată).
- ▶ Cu alte cuvinte, nu ni se poate atribui o nouă valoare.
- ▶ Valoarea variabilei finale trebuie assignată la momentul declarației în caz contrar, va da o eroare de compilare.

Final variable

- ▶ Un cuvânt cheie final poate fi aplicat variabilelor locale, variabilelor de instanță și variabilelor statice. În toate cazurile, face același lucru.
- ▶ Odată ce ați inițializat valoarea unei variabile finale, nu mai puteți atribui o nouă valoare.

Final method

- ▶ O metodă care este declarată cu cuvântul cheie final este cunoscută ca metodă finală în Java.
- ▶ O metodă finală nu poate fi suprascrisă în Java.
- ▶ Când o clasă este extinsă de o altă clasă, metoda sa poate fi suprascrisă pentru reutilizare, dar dacă vrem să împiedicăm o anumită metodă să fie suprascrisă, în acest caz, declarați această metodă ca fiind definitivă, deoarece o subclasă poate apela metoda finală a superclasei fără probleme, dar nu o poate suprascrie.
- ▶ Orice încercare de a face acest lucru va cauza o problemă de compilare

Final class

- ▶ O clasă care este declarată cu un cuvânt cheie final este cunoscută sub numele de clasă finală în Java.
- ▶ Clasa finală înseamnă restricționarea moștenirii !. Nu se poate moșteni de o altă clasă. Cu alte cuvinte, clasele Java declarate ca finale nu pot fi extinse (moștenite).
- ▶ O mulțime de clase în Java API sunt finale. De exemplu, clasa String este cel mai comun obiect predefinit din clasa finală în Java.

Final class

- ▶ Există două moduri de a face o clasă finală.
- ▶ 1. Primul mod de a face o clasă finală este să folosiți cuvântul cheie `final` în declarația clasei. Sintaxa pentru a face o clasă finală este următoarea: `public final class Student { // conținutul clasei }`
- ▶ 2. A doua modalitate este declararea tuturor constructorilor săi ca `private`. Dacă o clasă are numai constructori privați, aceasta nu poate avea subclasă.

Final class

- ▶ Există trei utilizări importante ale unei clase finale în Java.
- ▶ Acestea sunt după cum urmează:
- ▶ 1. Prima utilizare este de a preveni moștenirea, deoarece clasele finale nu pot fi extinse.
- ▶ 2. A doua utilizare este de a crea o clasă imutable ca și clasa predefinită String. Nu putem face o clasă imutable fără a o face definitivă.
- ▶ 3. O clasă finală este foarte utilă atunci când dorim securitate ridicată în orice aplicație, deoarece clasa finală nu poate fi extinsă.

Super keyword

- ▶ Cuvântul cheie **super** în Java este o variabilă de referință care se referă la un obiect de superclasă.
- ▶ Cu alte cuvinte, se referă la superclasa obiectului curent. Cuvântul cheie „super” intră în imagine cu conceptul de moștenire în Java.
- ▶ Cuvântul cheie „super” reprezintă întotdeauna un obiect al superclasei.
- ▶ Poate fi aplicat variabilelor, metodelor și constructorilor de superclasă.
- ▶ Permite utilizatorilor să acceseze membrii unei superclase dintr-o subclasă. Cu alte cuvinte, cuvântul cheie „super” poate fi utilizat pentru apelarea variabilei de instanță, a constructorului și a metodei superclasei imediate.

Super keyword

- ▶ Cuvântul cheie super Java poate fi utilizat în trei moduri:
 - ▶ 1.Pentru a accesa variabila de instanță a clasei părinte imediate.
 - ▶ 2.Pentru a apela constructorul clasei părinte imediate.
 - ▶ 3.Pentru a invoca metoda superclasei imediate.

Super keyword

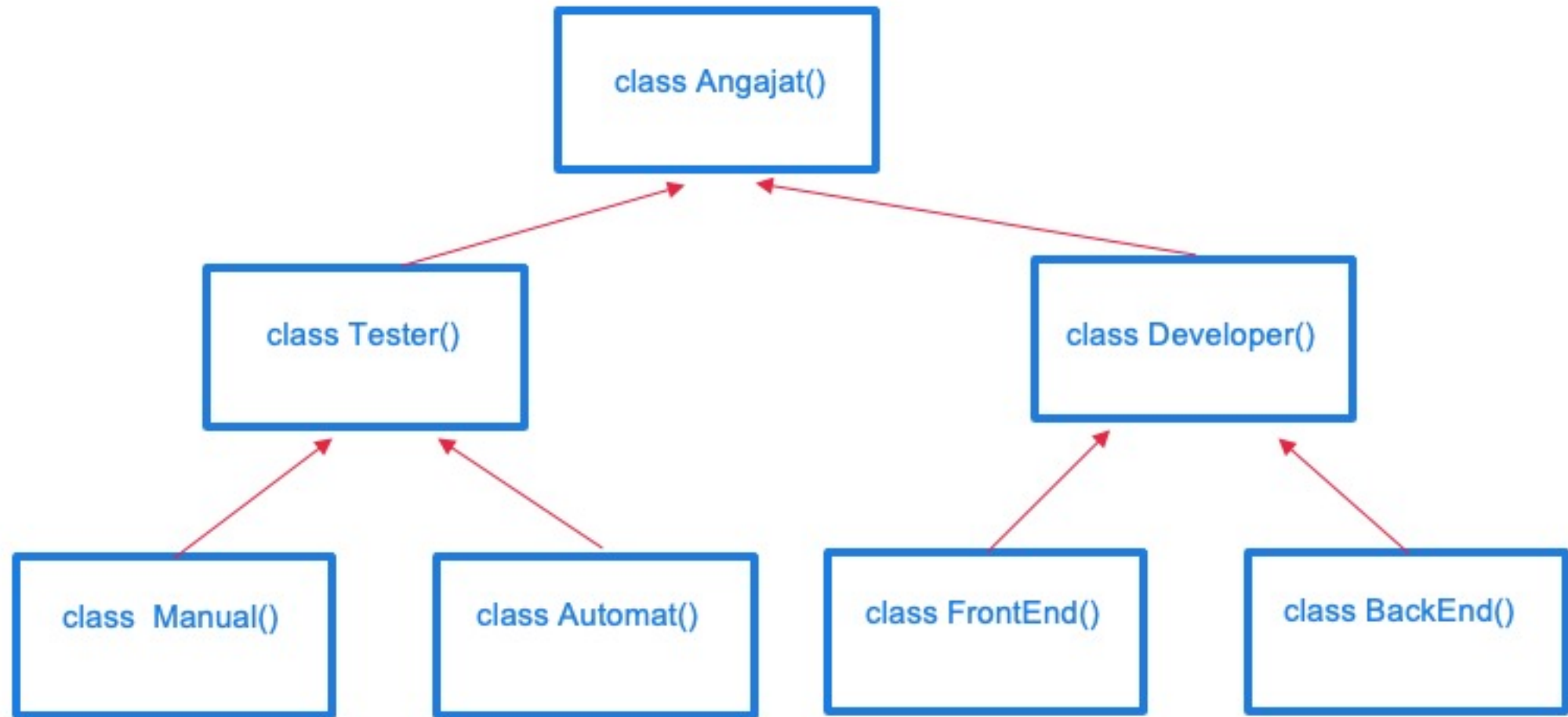
- ▶ Cuvântul cheie „super” poate fi folosit pentru a apela o variabilă de instanță a superclasei.
- ▶ Atunci când declarăm o variabilă de instanță din subclasă cu același nume ca și cea prevăzută în superclasă, nu putem accesa variabila de instanță a superclasei după numele acesteia din subclasă.
- ▶ Acest lucru se datorează faptului că are același nume.
- ▶ Pentru a rezolva această problemă, putem folosi cuvântul cheie super în subclasă pentru a ne referi la membrii superclasei.

Super keyword

- ▶ știm că un constructor este folosit pentru a crea o instanță a unei clase.
- ▶ Constructorul superclasei nu poate fi moștenit în subclasă.
- ▶ Poate fi apelat numai din constructorul subclasei folosind cuvântul cheie super. Cu alte cuvinte, constructorul superclasei poate fi invocat numai din interiorul constructorului subclasei.
- ▶ Dacă nu folosim cuvântul cheie super pentru a apela constructorul superclasei în constructorul oricărei clase, JVM va pune automat super-ul pe prima linie din constructorul clasei pentru a apela constructorul superclasei.


OOP : Inheritance

- ▶ Tehnica construirii unei noi clase utilizând o funcționalitatea unei clase existente se numește moștenire.
- ▶ Cu alte cuvinte, Moștenirea în Java este un proces în care o clasă copil dobândește toate proprietățile și comportamentele clasei părinte. Clasa existentă se numește clasă părinte, iar noua clasă se numește clasă copil.
- ▶ Moștenirea reprezintă relația IS-A, cunoscută și sub numele de relație părinte-copil.
- ▶ Este implementată în Java prin cuvinte cheie **extends** (pentru moștenirea clasei) și **implements** (pentru implementarea interfeței).





```
class Angajat {}
```

```
class Tester extends Angajat {}
```



“Tester extends Angajat”
inseamna
“Tetser IS A Angajat”

```
class Automat extends Tester {}
```



“Automat extends Tester”
inseamna
“Automat IS A Tester”

OOP : Inheritance

- ▶ O clasă părinte poate avea orice număr de clase derivate, dar o clasă derivată poate avea o singură clasă părinte.
- ▶ Prin urmare, Java nu acceptă moștenirea multiplă la nivelul clasei.

OOP : Inheritance

- ▶ Folosim moștenirea în java din următoarele motive:
 - ▶ Putem refolosi codul din clasa de bază.
 - ▶ Folosind moștenirea, putem crește caracteristicile clasei sau metodei prin suprascriere.
 - ▶ Este folosit pentru a obține polimorfismul în timpul rulării, adică suprascrierea metodei.
 - ▶ Folosind moștenirea, putem organiza informațiile într-o formă ierarhică.

OOP : Encapsulation

- ▶ Procesul de legare a datelor/atributelor și a metodelor corespunzătoare (comportament) împreună într-o singură unitate se numește încapsulare în Java.
- ▶ Cu alte cuvinte, încapsularea este o tehnică de programare care leagă membrii clasei (variabile și metode) împreună și împiedică accesul acestora de către alte clase, astfel putem păstra variabilele și metodele în siguranță împotriva interferențelor externe și a utilizării greșite.
- ▶ Fiecare clasă Java este un exemplu de încapsulare, deoarece scriem totul în clasă numai care leagă variabilele și metodele împreună și ascunde complexitatea acestora de alte clase.

OOP : Encapsulation

- ▶ În tehnica de încapsulare, declarăm câmpurile/atributele ca private în clasă pentru a împiedica alte clase să le acceseze direct.
- ▶ Datele încapsulate necesare pot fi accesate utilizând metodele publice Java getter și setter.
- ▶ Dacă câmpul este declarat privat în clasă, atunci acesta nu poate fi accesat de nimeni din afara clasei și ascunde câmpul în cadrul clasei. Prin urmare, se mai numește și ascunderea datelor (data hiding).