KeyTraining

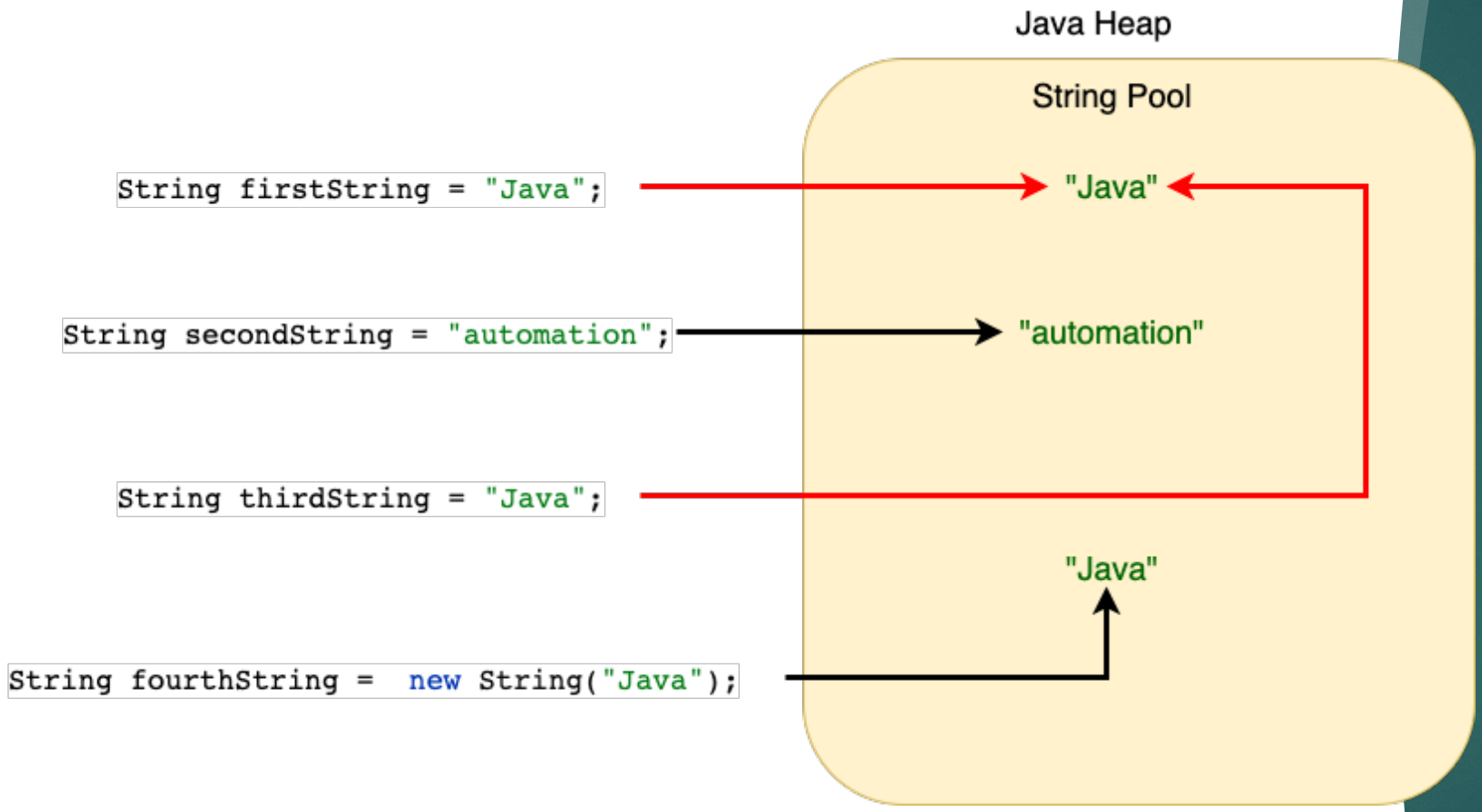
Curs java pentru testare automata

JAVA

Cuprins curs:

- String comparison
- Arrays
- For each

String comparison

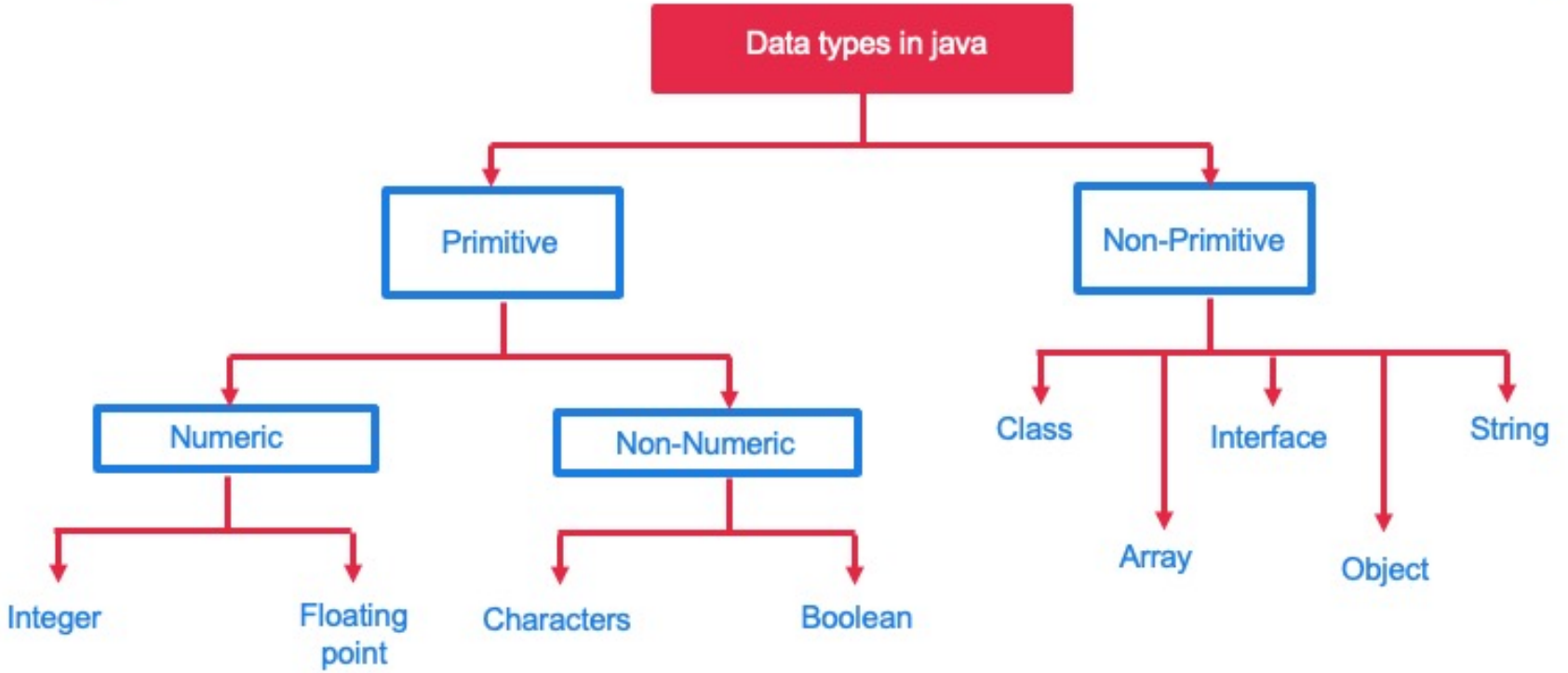


String Pool

- String Pool este o zonă de stocare în heap-ul Java. Alocarea Stringurilor, la fel ca orice alocare a obiectelor, se dovedește a fi o afacere costisitoare atât ca timp, cât și ca memorie.
- JVM efectuează câțiva pași în timp ce inițializează literele șirului pentru a crește performanța și a reduce cheltuielile de memorie.
- Pentru a reduce numărul de obiecte String create în JVM, clasa String păstrează un grup de stringuri. De fiecare dată când este creat un String nou, JVM verifică mai întâi String pool. Dacă Stringul există deja în grupul de Stringuri, atunci se intoarce o referință la obiect a acelui string.
- Dacă șirul nu există în pool, un nou obiect String se inițializează și este plasat în pool.

Data Types

- In Java tipurile de date sunt împărțite în două categorii după cum urmează:
- **Primitive data types**: Tipuri de date primitive (numite și tipuri intrinseci sau încorporate)
- **Non-primitive data types**: Tipuri de date neprimitive (numite și tipuri de date derivate sau de referință)


Data types in java
Primitive
Non-Primitive
Numeric
Non-Numeric
Class
Interface
String
Integer
Floating point
Characters
Boolean
Array
Object

Non - Primitive data types (Referinta)

- Array
- Un Array în java este un obiect care este utilizat pentru a stoca mai multe variabile de același tip.
- Aceste variabile pot fi tip de date primitive sau neprimitive.
- Exemplul declarării unei variabile Array de tip primitiv de date int este următorul:
- int [] note;
- Exemplul declarării unei variabile matrice de tip non-primitiv de date este :
- Tester [] testers; // Testser este un nume de clasă.

Array

- Array sunt obiecte care stochează mai multe variabile de același tip.
- Poate conține tipuri primitive, precum și referințe la obiecte(non primitive).
- Deoarece Array sunt obiecte, acestea sunt create în timpul rulării.
- Lungimea Array -ului este fixă.

Array

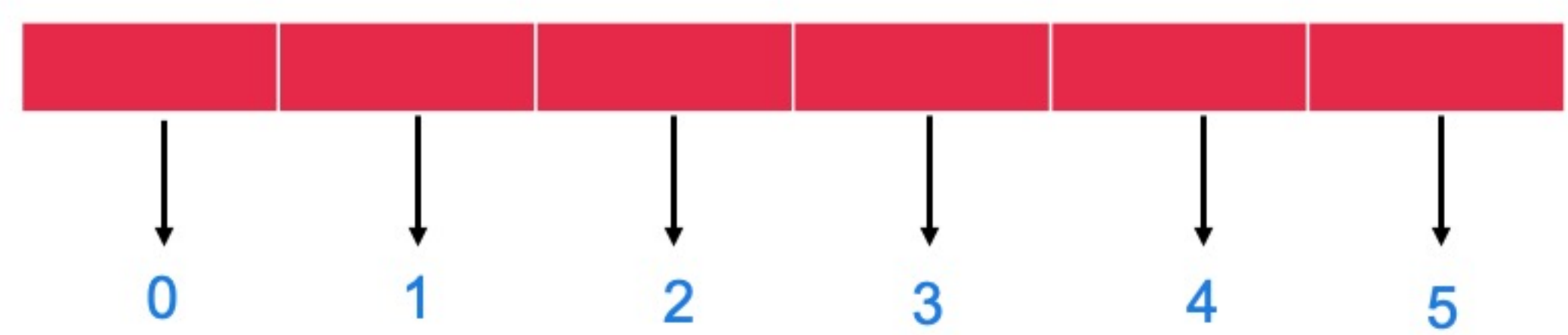
- Declarația Array-ul indică tipul elementului pe care Array-ul îl deține, urmat de identificator și acolade pătrate care indică faptul că identificatorul este tipul Array.

```
int [] cursanti = new int[6];
```

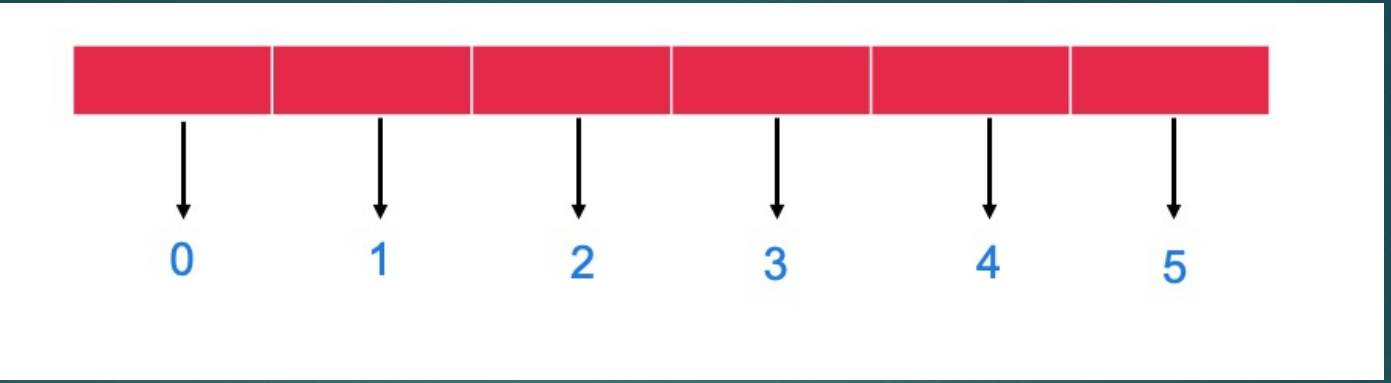
Array

```
int [] cursanti = new int[6];
```

- Există 6 sloturi aici, deoarece aceasta este lungimea specificată de noi; și fiecare dintre ele este capabil să dețină o singură valoare.



Array



- Aceste numere din partea de jos reprezintă indecsi fiecărui spațiu din Array. Este ca o adresă.
- Fiecare index indică unul dintre sloturi.
- Deci, dacă ar fi să avem această matrice de cursanti, care conține șase numere, fiecare dintre aceste numere ar fi un element din interiorul Array-ul.
- Observați că indecsii încep cu 0; iar indexul 0 reprezintă primul element al matricei.

Searching inside Arrays -Căutare secvențială

- Prima abordare este utilizarea unui algoritm de căutare secvențială. Algoritmul de căutare secvențială caută fiecare element din matrice până când găsește valoarea pe care o caută. Sau, până când ajunge la sfârșitul matricei.
- Acest algoritm este în regulă pentru Array-urile mici, dar este ineficient pentru Array-urile cu mai mult de câteva mii de elemente.

Searching inside Arrays -Binary Search

- A doua abordare este căutarea binară. Pentru căutarea binară, matricea trebuie mai întâi sortată.
- Verificarea începe cu elementul din mijlocul matricei, pentru a vedea dacă este egal, mai mare sau mai mic decât valoarea pe care o căutam.
- Dacă este mai mare decât, atunci nu este nevoie să mai căutam in a doua jumătate a matricei. Pentru că știm că totul acolo va fi mai mare.
- La fel, dacă numărul de mijloc este mai mic decât valoarea pe care o căutam, atunci nu este nevoie să căutam in prima jumătate a matricei.

Searching inside Arrays -Binary Search

- Dacă valoarea este egală, ai noroc și ai terminat.
- Dacă nu este egala, ai eliminat încă jumătate din matrice, unde știi că valoarea nu este, și apoi continui cu o altă iterație.
- Fiecare iterație continuă astfel - eliminând jumătate din matrice pentru ceea ce trebuie căutat, până când găsiți valoarea sau căutați întreaga matrice.
- Aceasta este o abordare mult mai rapidă decât căutarea secvențială.

For Each (Enhanced loop)

- For each loop din Java (denumit și enhanced loop) este o caracteristică extinsă a limbajului Java care a fost introdusă odată cu versiunea J2SE 5.0.
- Această caracteristică este special concepută pentru a recupera elemente ale unei matrice în mod mai eficient decât să se utilizeze indexuri in accesarea elementelor din matrice.
- For each loop poate fi, de asemenea, utilizat pentru a prelua elemente ale unei colecții. O colecție reprezintă un grup de elemente precum valori întregi, șiruri sau obiecte.
- Enhanced for loop execută în mod repetat un grup de instrucțiuni pentru fiecare element al colecției.

For Each (Enhanced loop)

- Syntaxa:
- for(Type Identifier : Expression) {
- // statements; }
- Aici, Type reprezintă tipul de date sau obiectul utilizat;
- Identificatorul specifică numele unei variabile de iterație care primește elemente dintr-o colecție, unul câte unul, de la început până la sfârșit;
- Expresie este un obiect al interfeței java.lang.Iterable sau o matrice.