KeyTraining

Curs java pentru testare automata

JAVA

Studentul poate utiliza prezentul material si informatiile continute in el exclusiv in scopul asimilarii cunostintelor pe care le include, fara a afecta dreptul de proprietate intelectuala detinut de Key Talents Srl

KeyTraining

Cuprins curs:

- Intro git
- Ce este Java
- Istoria Java
- Cum functioneaza Java
- Pachete in Java
- Introducere in Clase si obiecte

Version Control

- Controlul versiunilor reprezinta gestionarea modificărilor documentelor de-a lungul timpului.
- Urmărind și înregistrând modificările pe care le aduceți fișierului sau seturilor de fișiere de-a lungul timpului, un sistem de control al versiunilor vă oferă puterea de a revizui sau chiar de a restabili versiunile anterioare.
- Controlul versiunilor realizează instantanee ale fiecărei revizuiri a proiectului dvs.
- Apoi, puteți accesa aceste versiuni pentru a le compara sau restabili după cum este necesar.

KeyTraining

Version Control

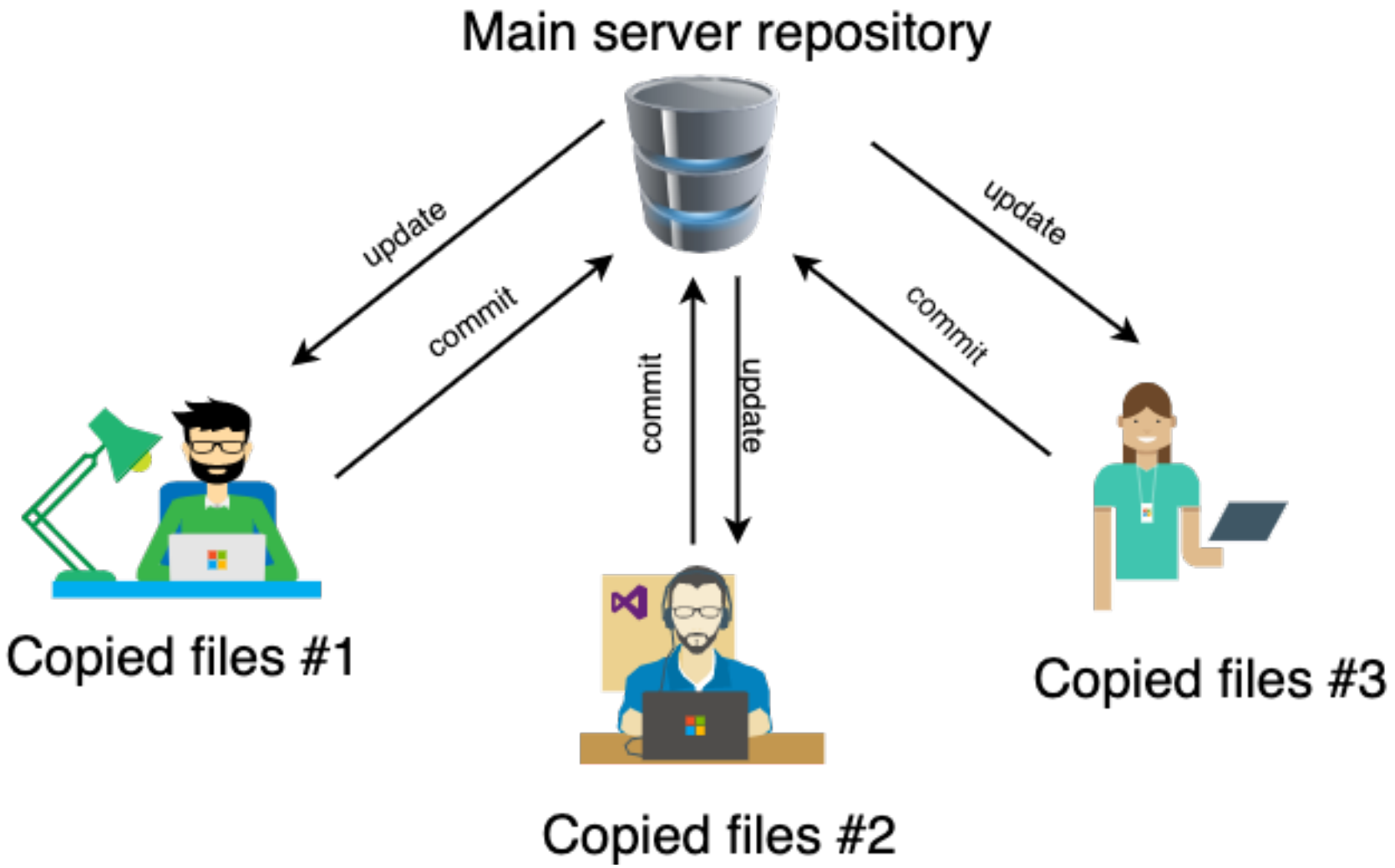
- Cea mai obișnuită utilizare a unui control de versiune este atunci când copiem fișiere într-un alt folder sau redenumim fișierele noastre cu vs1..vsN și
- CVCS – Centralised Version Control Sistem
- DVCS – Distributed Version Control Sistem

CVCS

- În CVCS, utilizatorii pot actualiza simultan modificările la un server central.
- Toate datele sunt păstrate la un singur server central.
- Beneficiile includ:
  - În același timp, mai mulți utilizatori pot lucra într-o singură locație de stocare.
  - Posibilitatea de a controla, cine poate modifica sau cine poate doar vizualiza, adică oferă drepturi de gestionare a codului sursa.
- Dezavantajele includ:
  - Dacă serverul central nu poate fi accesat din varii motive, întreaga echipa de utilizatori trebuie să aștepte până când revine la starea sa inițială.
  - Dacă hard disk-ul central al sistemului este deteriorat, dacă nu a existat nicio copie de rezervă, exista riscul ca tot codul sa se piardă.

KeyTraining

Centralized Version Control



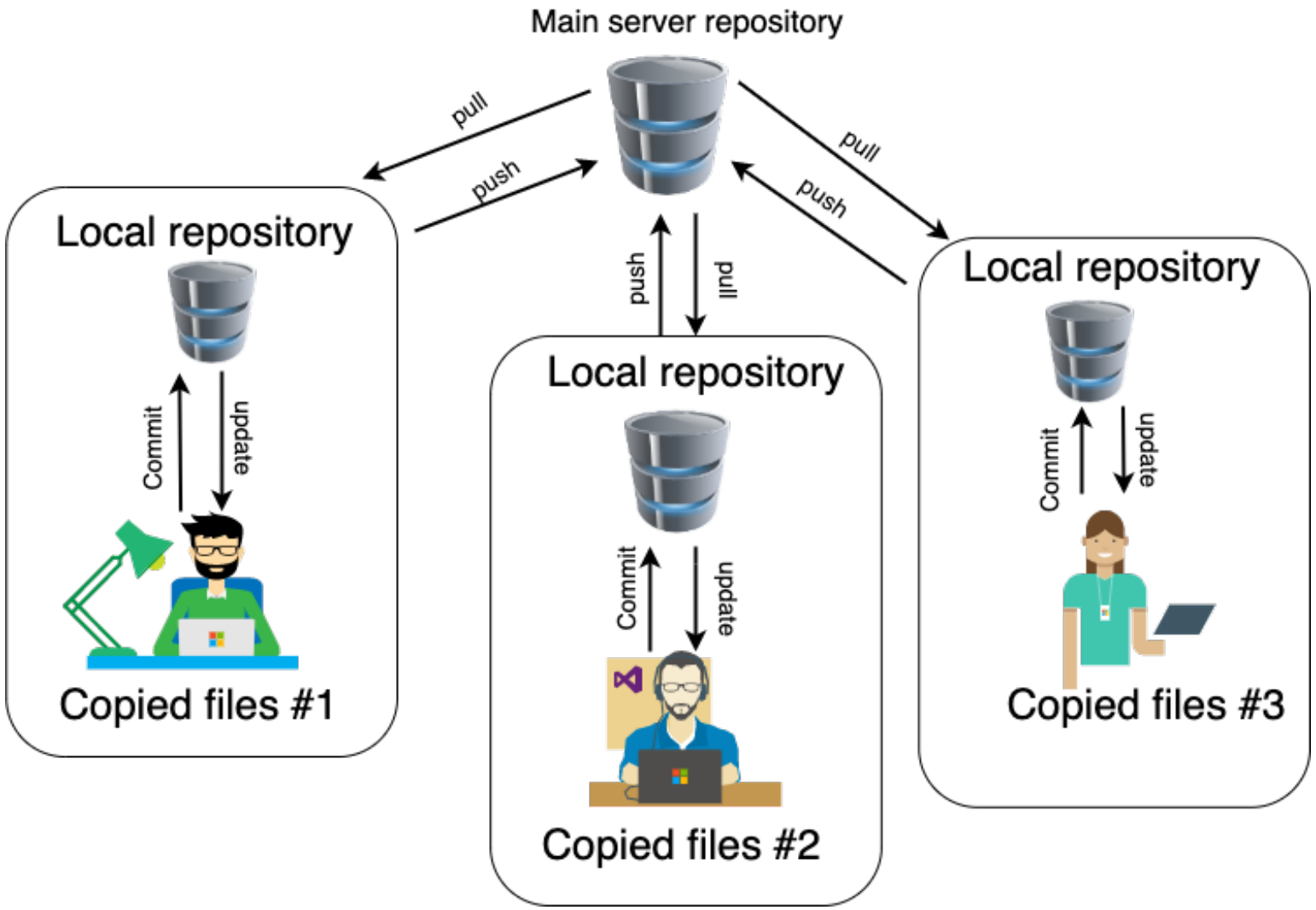
KeyTraining

DVCS

- În DVCS, datele sunt copiate nu numai într-o bază de date centrală a serverului, ci și între baza de date locală a clienților, adică atunci când un client se conectează la main repository, el obține nu numai diferențele dintre versiuni, ci toate datele sunt copiate în baza sa de date locală.
- S-ar putea spune că DVCS este o combinație de CVCS și LVCS. DVCS-urile populare sunt Git sau Mercurial.
- Riscul de pierdere a datelor a fost redus la minimum.

KeyTraining

Distributed Version Control



KeyTraining

GIT

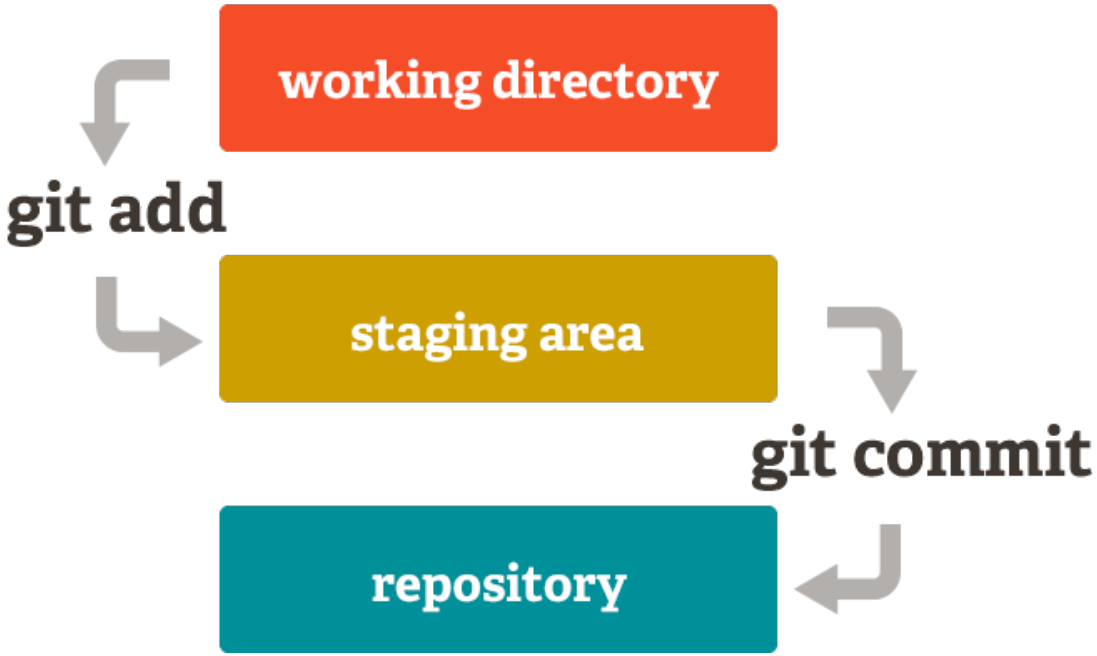
- Git este un sistem de control al versiunilor (DVCS = distributed version control sistem)care vă permite gestionarea și să urmărirea istoricului codului sursă.
- Git a fost creat de Linus Torvalds în 2005 pentru dezvoltarea kernel-ului Linux, alți dezvoltatori de kernel contribuind la dezvoltarea sa inițială.
- Git este software gratuit și open source distribuit în condițiile licenței publice generale GNU versiunea 2.

GIT vs GITHUB

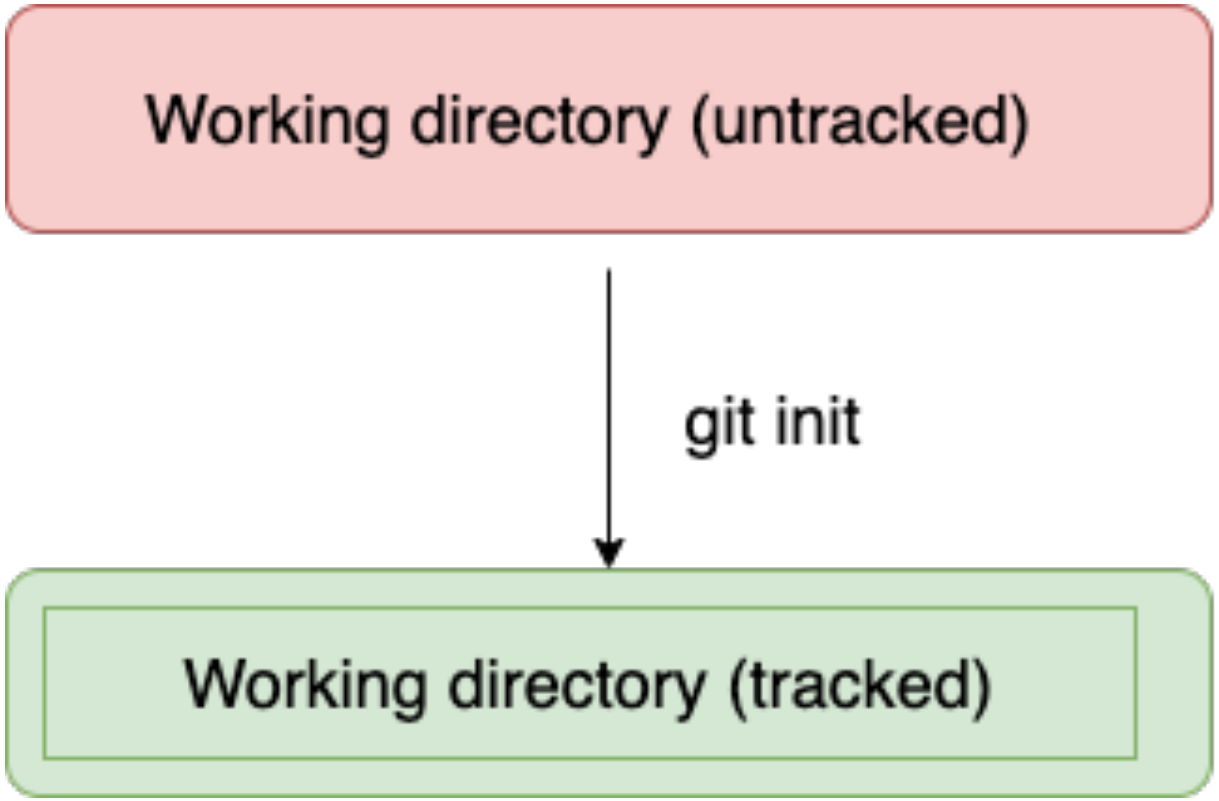
- GitHub este un serviciu de găzduire cloud care permite să gestionați Git repositories.
- Și ce este mai exact un serviciu de găzduire cloud pentru Git?
- Este o bază de date online care permite să urmărim și să partajam proiectele în afara computerului / serverului local.
- GitHub, GitLab, BitBucket

GIT – the 3 stages

- Working directory
- Staging area (index)
- Repository

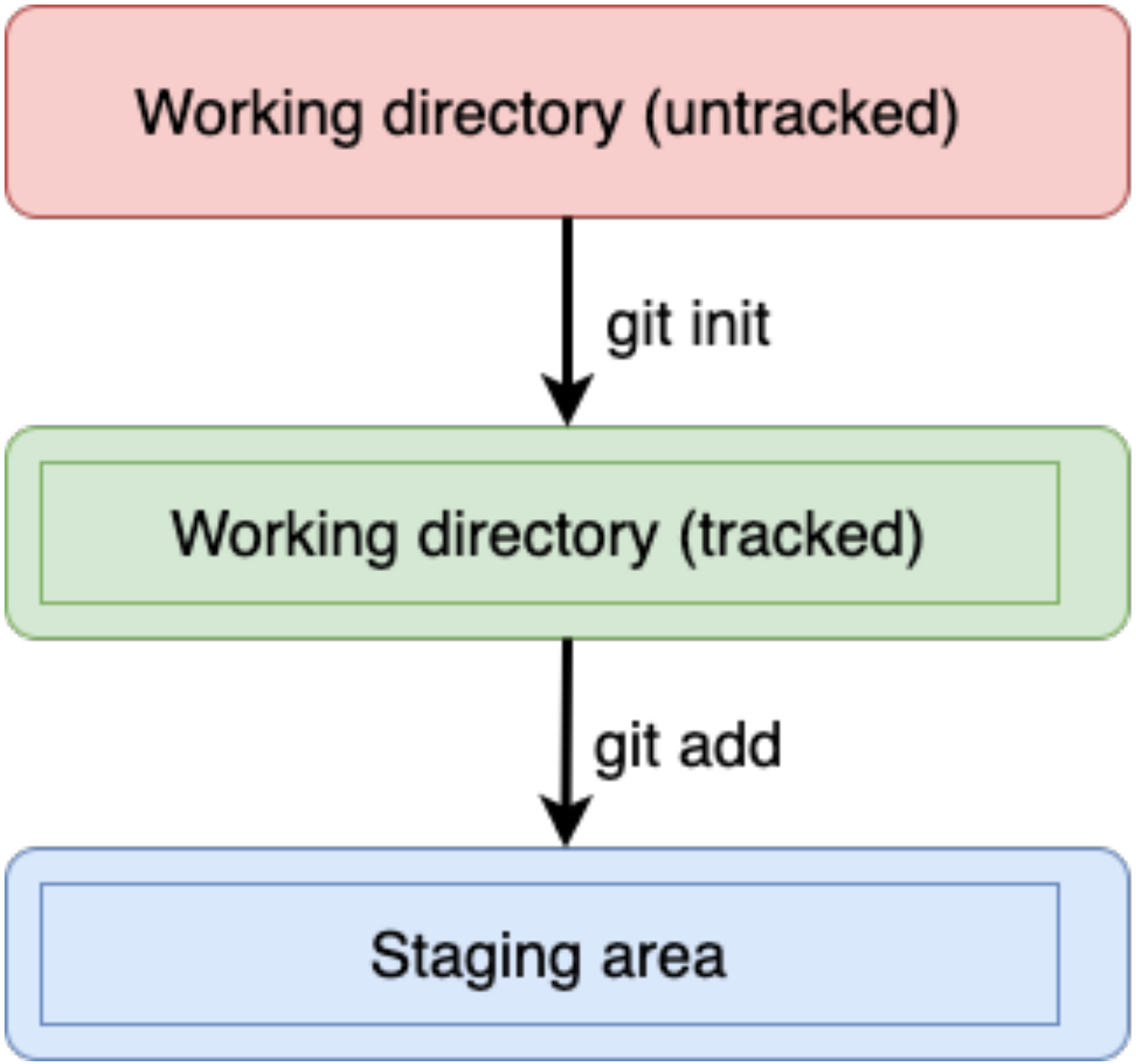


KeyTraining



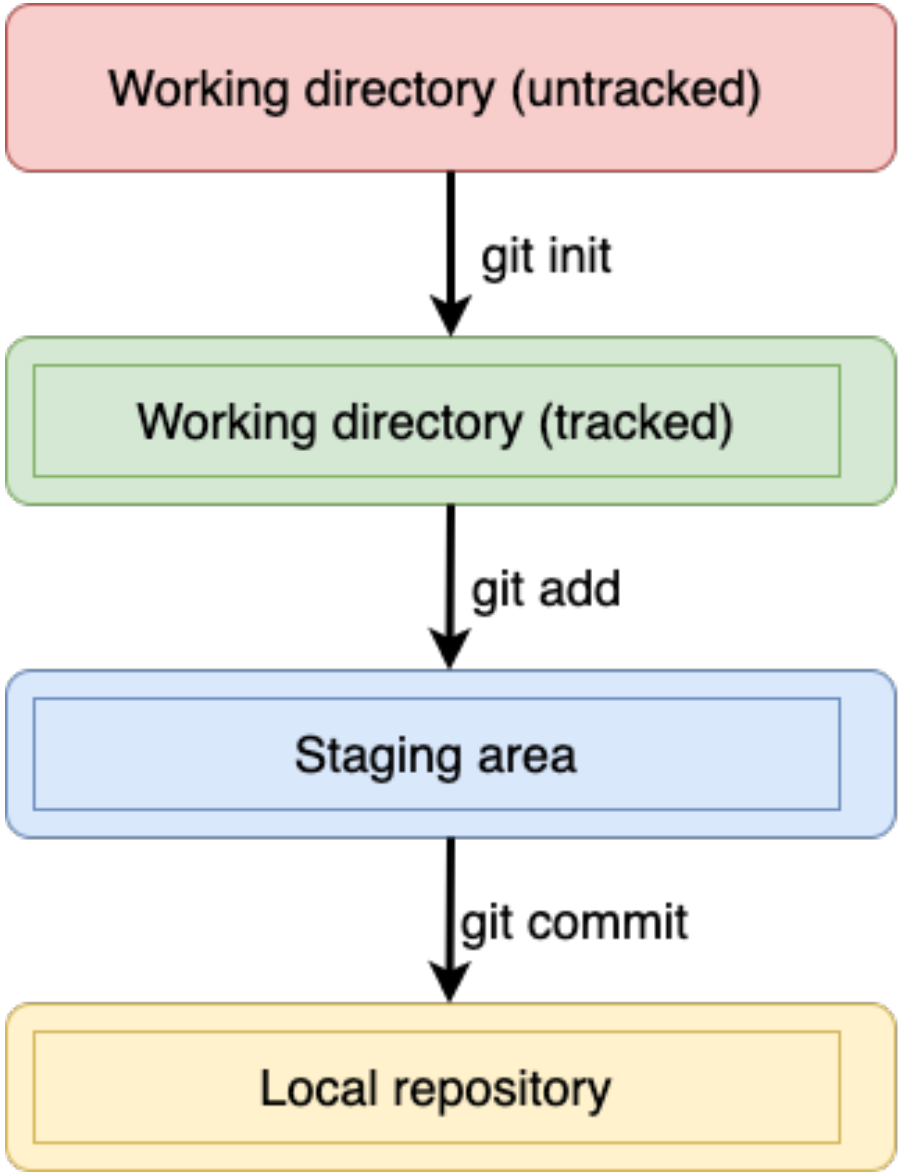
Git stages

KeyTraining



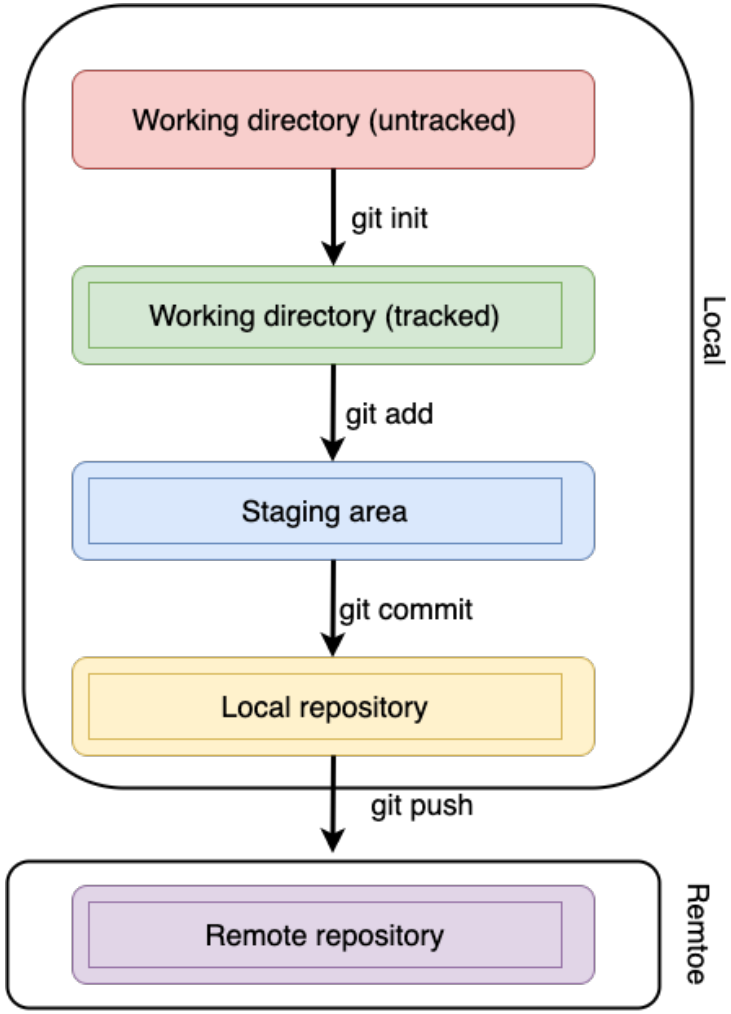
Git stages

KeyTraining



Git stages

KeyTraining



Git stages

KeyTraining

KeyTraining

Java

- Java este un limbaj de programare. A fost dezvoltat și lansat de Sun Microsystems (în prezent Oracle Corporation) în 1995. A fost dezvoltat de James Gosling.
- Java a devenit foarte popular mai ales datorită independenței sale de platforma pe care rulează. Un program Java poate rula pe arhitecturi multiple.
- Programul o dată ce îl compilați, va genera un cod care nu este procesat de procesorul gazdă. Acesta va fi procesat de un procesor virtual Java, care se găsește în mașina virtuală Java (JVM). Acest JVM va traduce codul respectat în instrucțiuni care sunt trimise procesorului mașinii gazdă.

KeyTraining

Java

- Java are o versiune JVM pentru fiecare sistem de operare, astfel codul va fi tradus de același procesor virtual Java pentru fiecare tip de arhitectură eliminând independența platformei.
- Astfel, aplicația Java respectă principiile WORA (Write Once Run Anywhere).

KeyTraining

Istoria Java

- În 1990, Sun Microsystems Inc. (SUA) și-a imaginat un proiect de dezvoltare a software-ului pentru dispozitive electronice de consum care ar putea fi controlat de o telecomandă.
- Inițial, acest proiect a fost numit Stealth Project, dar mai târziu numele său a fost schimbat în Green Project.
- În ianuarie 1991, Bill Joy, James Gosling, Patrick Naughton, Mike Sheridan și alți câțiva s-au întâlnit în Aspen, Colorado, pentru a discuta despre acest proiect.
- Slujba lui Mike Sheridan a fost să se concentreze asupra dezvoltării afacerilor. Patrick Naughton urma să lucreze la sistemul grafic. James Gosling urma să recunoască limbajul de programare adecvat proiectului.
- El a crezut că limbajele de programare C și C ++ ar putea fi folosite pentru a dezvolta acest proiect.

KeyTraining

Istoria Java

- Dar problema cu care s-a confruntat în utilizarea acestor limbaje este că C și C ++ erau limbaje de programare dependente de sistem.
- Datorită cărora nu au putut fi utilizate pe diferite sisteme de calculatoare sau dispozitive electronice. Așadar, a început să dezvolte un nou limbaj de programare care să fie complet independent de platformă și să poată fi rulat pe orice dispozitiv electronic.
- Acest limbaj de programare a fost numit inițial Oak, dar ulterior a fost schimbat în Java în 1995.
- Astfel, Java a fost dezvoltat de James Gosling, Patrick Naughton, Chris Warth, Ed Frank și Mike Sheridan la Sun Microsystems în 1991.
- A fost nevoie de 18 luni pentru a dezvolta prima versiune de lucru a Java.

KeyTraining

Istoria Java

- Lista versiunilor Java pana acum

KeyTraining

De ce se cheama Java?

- James Gosling și membrii echipei sale consumau multa cafea în timp ce dezvoltau limbajul de programare Java.
- Au simțit că au dezvoltat un limbaj de programare mai bun din cauza cafelei de bună calitate pe care au consumat-o.
- Prin urmare, au păstrat numele pentru limbaj ca Java. Deoarece Java era un nume atât de unic, majoritatea membrilor echipei au preferat să il denumească Java.
- Java este o insulă din Indonezia unde cafeaua a fost numită cafea java. Numele Java a fost ales de James Gosling în timp ce consuma cafea lângă biroul său în timpul dezvoltării limbajului Java.
- Astfel, simbolul pentru Java este ceașca de cafea.

KeyTraining

Java si legatura cu C/C++

- După cum știm, era modernă a programării a început cu C, apoi sa mutat în C ++ și apoi în Java.
- C și C ++ sunt printre cele mai importante limbaje de programare computerizate inventate vreodată și sunt încă utilizate pe scară largă astăzi în lume.
- Java își moștenește sintaxa din limbajul C.
- Mai multe caracteristici orientate obiect ale Java au fost influențate de C ++.
- La momentul creării Java, mulți programatori erau familiarizați cu sintaxa C / C ++.
- De aceea, Java a folosit o sintaxă similară legată de C / C ++, astfel încât programatorilor C / C ++ să le fie relativ ușor să învețe Java.
- C, C ++ și Java definesc un cadru comun și conceptual pentru programatorul profesionist, astfel încât programatorii să nu se confrunte cu probleme majore în timp ce trec de la un limbaj la altul.

KeyTraining

Java si legatura cu C/C++

- Dar Java nu este o versiune îmbunătățită a C ++
- Nu este nici în sus, nici în jos compatibil cu C ++
- Mai mult, Java nu a fost conceput pentru a înlocui C ++
- Java a fost conceput și dezvoltat pentru a rezolva un anumit set de probleme care au apărut în timpul proiectului.
- Atât C ++ cât și Java vor coexista timp de mulți ani.

KeyTraining

Java si legatura C#

- C # a fost dezvoltat de Microsoft pentru a sprijini .NET Framework.
- Este strâns legat de Java. De exemplu, ambele definesc aceeași sintaxă generală, acceptă programarea distribuită și utilizează același model de obiect.
- Desigur, există multe diferențe între Java și C #, dar aspectul general al acestor limbi este foarte asemănător.

KeyTraining

Caracteristicile Java

- Știm că Java este un limbaj orientat pe obiecte și independent de platformă.
- Pe lângă aceste două cuvinte cheie populare (caracteristici), există multe alte cuvinte cheie pe care trebuie să le cunoaștem.
- Acest lucru se datorează faptului că istoria Java nu se poate completa fără a arunca o privire asupra caracteristicilor Java.
- Prin urmare, este important să cunoașteți cuvintele cheie Java (adică caracteristicile Java), deoarece acestea au jucat roluri importante în modelarea limbajului Java.

KeyTraining

Caracteristicile Java

- Simple
- Object-oriented
- Distributed
- Robust
- Secure
- System independence
- Portability
- Interpreted
- High performance
- Multi threaded
- Dynamic

KeyTraining

Caracteristicile Java -Simple

- Java este conceput ca un limbaj de programare simplu.
- Când Java a fost dezvoltat, echipa Java a dorit să fie simplu pentru ca programatorul profesionist să învețe și să îl utilizeze eficient.
- Acest lucru se datorează faptului că trebuia să funcționeze pe diferite dispozitive electronice sau pe mașini mici unde este disponibilă mai puțină memorie.
- În primul rând, mai multe caracteristici dificile, stângace și confuze ale altor limbaje de programare, cum ar fi C și C ++, au fost omise în Java.
- De exemplu, caracteristica indicilor (pointers) care este foarte dificilă atât pentru cursanți, cât și pentru programatori.

KeyTraining

Caracteristicile Java -Simple

- Există în principal trei motive pentru care au eliminat conceptul de pointer din Java.
- Acestea sunt după cum urmează:
- 1. Conceptul de Pointer poate duce la confuzie pentru un programator.
- 2. Pointerul poate strica cu ușurință un program. De exemplu, când adăugăm doi pointeri, programul se poate bloca imediat.
- 3. Utilizarea funcției Pointer poate afecta securitatea, deoarece programele dăunătoare precum virusii și alte programe de hacking pot fi dezvoltate folosind pointers.
- Din motivele de mai sus, caracteristica pointerului a fost complet eliminată din tehnologia Java.

KeyTraining

Caracteristicile Java –Object-oriented

- Java este un limbaj de programare pur orientat pe obiecte (OOP).
- Aceasta înseamnă că programul Java este dezvoltat folosind clase și obiecte.
- Cu alte cuvinte, pentru a scrie un program în Java, avem nevoie de cel puțin o clasă sau un obiect.
- Acum, întrebarea care apare, ce este un obiect în Java?
- Un obiect este orice lucru care există în lumea reală. De exemplu, fiecare ființă umană, o carte, un copac, stilou, creion și așa mai departe. Fiecare obiect are proprietăți și prezintă un anumit comportament (acțiune).

KeyTraining

Caracteristicile Java –Object-oriented

- Luați în considerare un câine.
- Proprietățile unui câine sunt: numele, înălțimea, culoarea, vârsta etc.
- Aceste proprietăți sunt reprezentate de variabile în Java.
- Câinele obiect are mai multe comportamente (acțiuni), cum ar fi: alergatul, lătratul, mâncatul, somnul etc.
- Aceste comportamente sau acțiuni sunt reprezentate de diferite metode (funcții) în programarea Java.
- Deci, putem spune că un obiect conține variabile și metode.

KeyTraining

Caracteristicile Java –Object-oriented

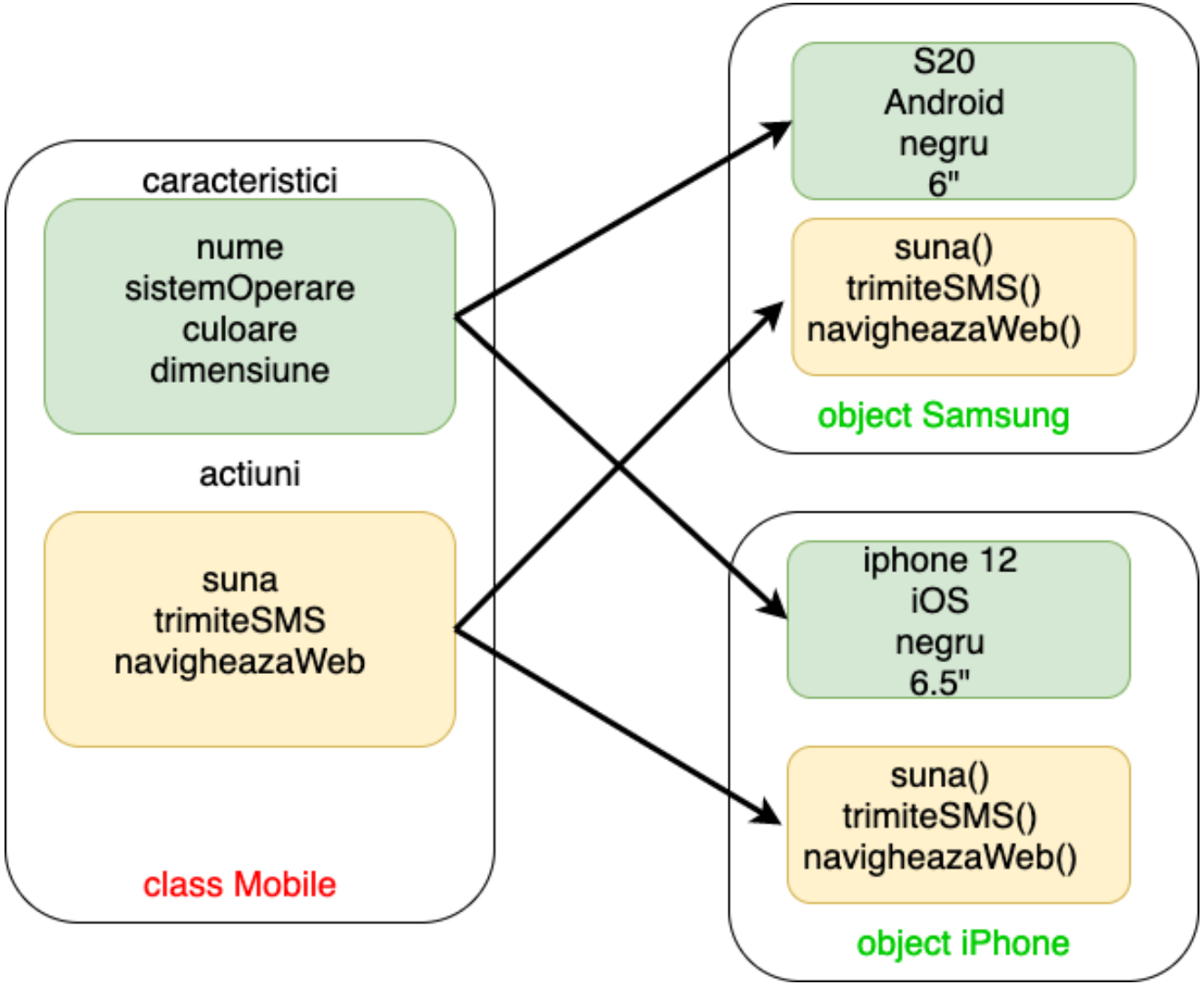
- A doua întrebare care apare: ce este o clasă în Java?
- O clasă definește numele unui grup dat mai multor obiecte având aceleași proprietăți și comportamente (acțiuni), sau o clasa poate fi considerata un template (sablon) pentru Obiect.
- Un telefon mobil poate fi Samsung, Huawei, sau Apple.
- Toate aceste trei telefoane mobile au aceleași comportamente precum mesageria, apelurile, navigarea etc.
- Prin urmare, vor face parte din același grup numit: Mobil.
- Deci, numele clasei este Mobil care conține trei obiecte.
- Cu alte cuvinte, o clasă în java poate fi definită ca un model sau un plan pentru crearea obiectelor.

KeyTraining

Caracteristicile Java –Object-oriented

- O clasă conține, de asemenea, proprietăți și acțiuni, adică variabile și metode precum obiecte.

KeyTraining



Caracteristicile Java –Object-oriented

KeyTraining

Caracteristicile Java –Distributed

- Java este conceput pentru a susține mediul distribuit al internetului, deoarece gestionează protocoalele TCP / IP și UDP.
- Folosind Java, putem crea un program pentru a obține informații și le putem distribui pe diferite computere dintr-o rețea.
- Java acceptă, de asemenea, funcția de invocare a metodelor la distanță (RMI -Remote Method Invocation), care permite unui program să distribuie metode într-o rețea.

KeyTraining

Caracteristicile Java –Robust

- Înțelesul lui Robust este „puternic”.
- Aceasta înseamnă că programele Java sunt puternice, deoarece nu se blochează ușor ca programele C sau C ++.
- Există în principal trei motive pentru a fi robust.
- În primul rând, Java are o caracteristică excelentă încorporată numită gestionarea excepțiilor. O excepție este o stare anormală sau o eroare care apare în timpul rulării.
- Dacă apare o excepție într-un program java, programul se încheie brusc și apar probleme cum ar fi pierderea datelor.
- Pentru a depăși astfel de tipuri de probleme, putem folosi mecanismul de gestionare a excepțiilor. Aceasta înseamnă că, deși are loc o excepție într-un program, nu se va produce nici un prejudiciu (adică pierderea datelor).

KeyTraining

Caracteristicile Java –Robust

- Al doilea motiv este că Java are o caracteristică robustă de gestionare a memoriei. Majoritatea programelor C și C ++ se blochează din cauza nealocării unui spațiu suficient în memorie pentru un program.
- Astfel de probleme nu vor apărea în Java, deoarece nu este necesar să alocăm sau să de-alocăm memoria în Java. Totul este îngrijit de JVM.
- De exemplu, JVM alocă memoria necesară de către un program Java.
- Al treilea motiv este că Java acceptă o caracteristică de verificare a erorilor în diferite etape: verificarea timpurie în timpul compilării și verificarea dinamică în timpul rulării.

KeyTraining

Caracteristicile Java –Secure

- Deoarece Java acceptă mediul distribuit al internetului, acesta oferă de asemenea mai multe funcții de securitate.
- Problemele de securitate, cum ar fi falsificarea, uzurparea identității și amenințările cu viruși, pot fi eliminate sau reduse prin utilizarea Java in aplicațiile expuse pe Internet.
- Programul Java rulează întotdeauna în mediul de execuție Java cu interacțiune aproape nulă cu sistemul de operare al sistemului pe care ruleaza, prin urmare este mai sigur.

KeyTraining

Caracteristicile Java –System independence

- Compilatorul Java compilează codul sursă în cod de octeți(bytes), care este independent de orice arhitectură a mașinii pe care rulează.
- Cu alte cuvinte, codul de octeți al programului Java nu depinde de computerul pe care rulează.
- Poate fi rulat pe orice computer de sistem cu orice procesor și orice sistem de operare care implementează Java Virtual Machine.

KeyTraining

Caracteristicile Java –Portability

- Dacă un program dă același rezultat pe fiecare sistem pe care rulează fără modificări, acel program se numește portabil.
- Din acest punct de vedere programele Java sunt portabile.

KeyTraining

Caracteristicile Java –Interpreted

- În timpul compilării, compilatorul Java convertește codul sursă al programului în cod de octeți(bytes).
- Acest cod de octeți poate fi executat pe orice sistem cu ajutorul unui interpretor Java în cadrul JVM.
- Dacă luăm orice alt limbaj de programare, numai un compilator sau un interpretor este folosit pentru a rula programele.
- Dar în Java, folosim atât compilatorul, cât și interpretorul pentru executarea programului.

KeyTraining

Caracteristicile Java –High Perfomance

- Viteza interpretorului în interiorul JVM pentru a executa un program este lentă.
- Pentru a depăși această problemă, echipa Java a introdus un compilator JIT (Just In Time – JIT Compiler) care îmbunătățește performanța de interpretare a codului prin cache.
- Datorită acestui fapt, viteza de execuție a programului Java este îmbunătățită.
- Deci, atât interpretul, cât și compilatorul JIT din JVM lucrează împreună pentru a rula programul.

KeyTraining

Caracteristicile Java –Multithreaded

- Java acceptă programarea multithreading care permite scrierea de programe pentru a procesa mai multe seturi de instrucțiuni simultan.
- Un thread este un proces individual pentru a executa un grup de instrucțiuni.
- JVM utilizează mai multe threadu-uri pentru a executa diferite blocuri de cod.
- Crearea mai multor thread-uri se numește „multithread” în Java

KeyTraining

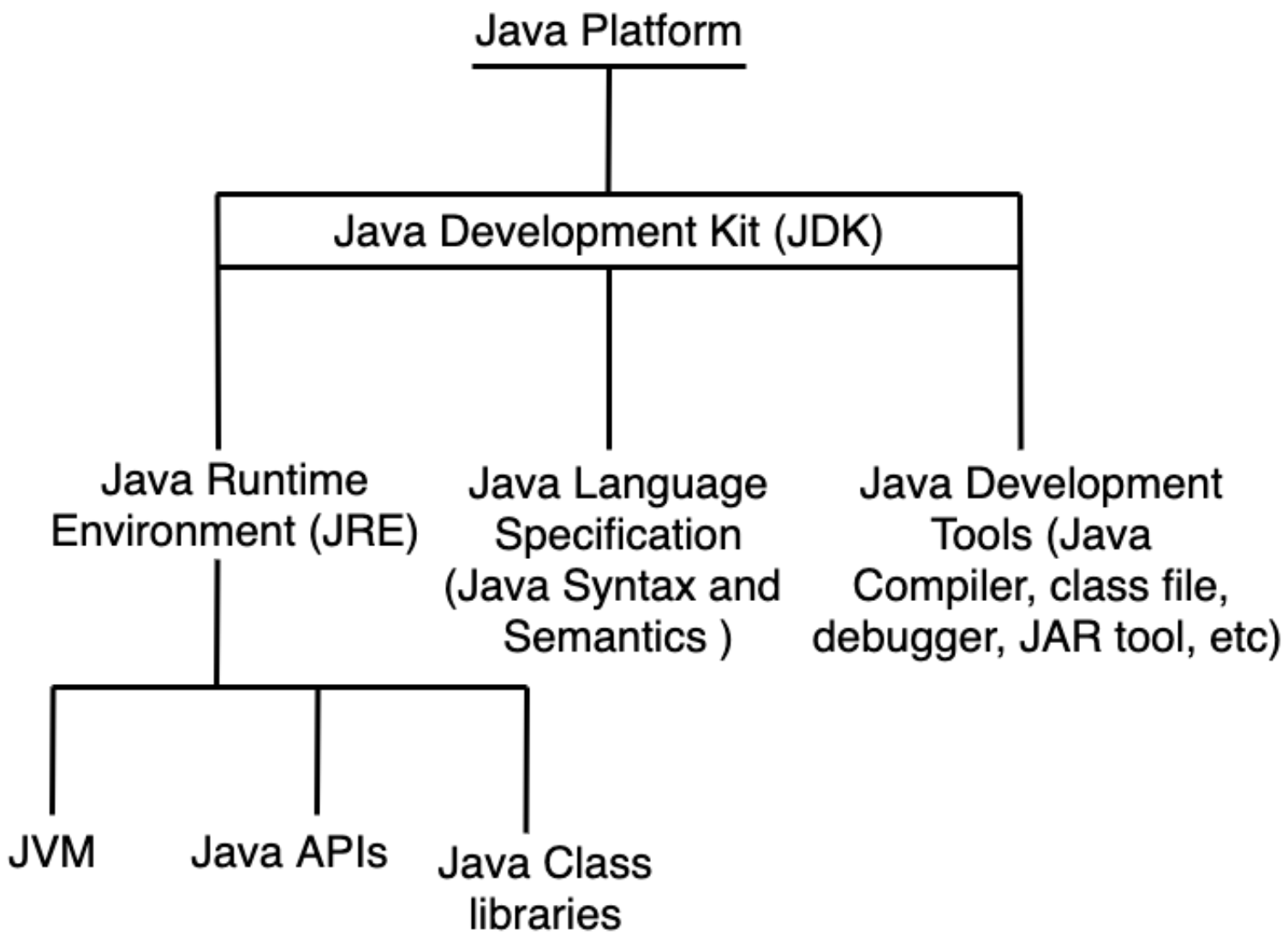
Caracteristicile Java –Dynamic

- Programele Java pot transporta o cantitate extinsă de informații în timp de execuție(runtime) care pot fi utilizate pentru a verifica și rezolva accesele la obiecte în timpul execuției.

KeyTraining

JAVA Platform

- O platformă este un mediu software sau hardware în care se execută sau rulează un program.
- Unele dintre cele mai cunoscute platforme sunt Microsoft Windows, Linux, Mac.
- Platforma Java este doar o platformă bazată pe software care rulează pe alte platforme bazate pe hardware, cum ar fi Microsoft Windows, Linux și Mac.
- Platforma Java este compusă dintr-o ierarhie de componente care este prezentată în figura de mai jos.



KeyTraining

Java Platform

KeyTraining

Java JDK (java development kit)

- Java Development Kit (JDK) este un mediu de dezvoltare software dezvoltat și distribuit de Oracle.
- Este utilizat pentru construirea (dezvoltarea) de aplicații software și applet-uri Java.
- În cuvinte simple, este folosit pentru a scrie programe java de către programatori.
- Java Development Kit are următoarele componente, descrise mai jos.

KeyTraining

Java JRE (java runtime environment)

- Java Runtime Environment (JRE): este un mediu necesar pentru a rula (executa) o aplicație java.
- Nu poate fi folosit pentru dezvoltarea aplicațiilor Java.
- Este utilizat numai pentru executarea programului.
- Java Runtime Environment este o colecție de programe care conține :
- JVM (Java Virtual Machine)
- Java API (Application Programming Interface)
- Java Class Libraries care ajută la dezvoltarea programelor de aplicații Java.

KeyTraining

Java language specification

- Java Language Specification reprezinta o definiție tehnică a sintaxei și semanticii limbajului de programare Java.
- Aici, puteți găsi specificațiile complete ale limbajului Java la https://docs.oracle.com/javase/specs/

KeyTraining

Java Development Tools

- Java Development Tools: constă din Java compiler, JAR tool, class file disassembler, debugger, JRE builder, etc.
- Java Development Tools oferă totul pentru compilarea, rularea, monitorizarea, debugging-ul și documentarea aplicațiilor.
- Cele mai importante instrumente sunt javac compiler, java launcherși instrumentul de documentare javadoc.

KeyTraining

Java Compiler

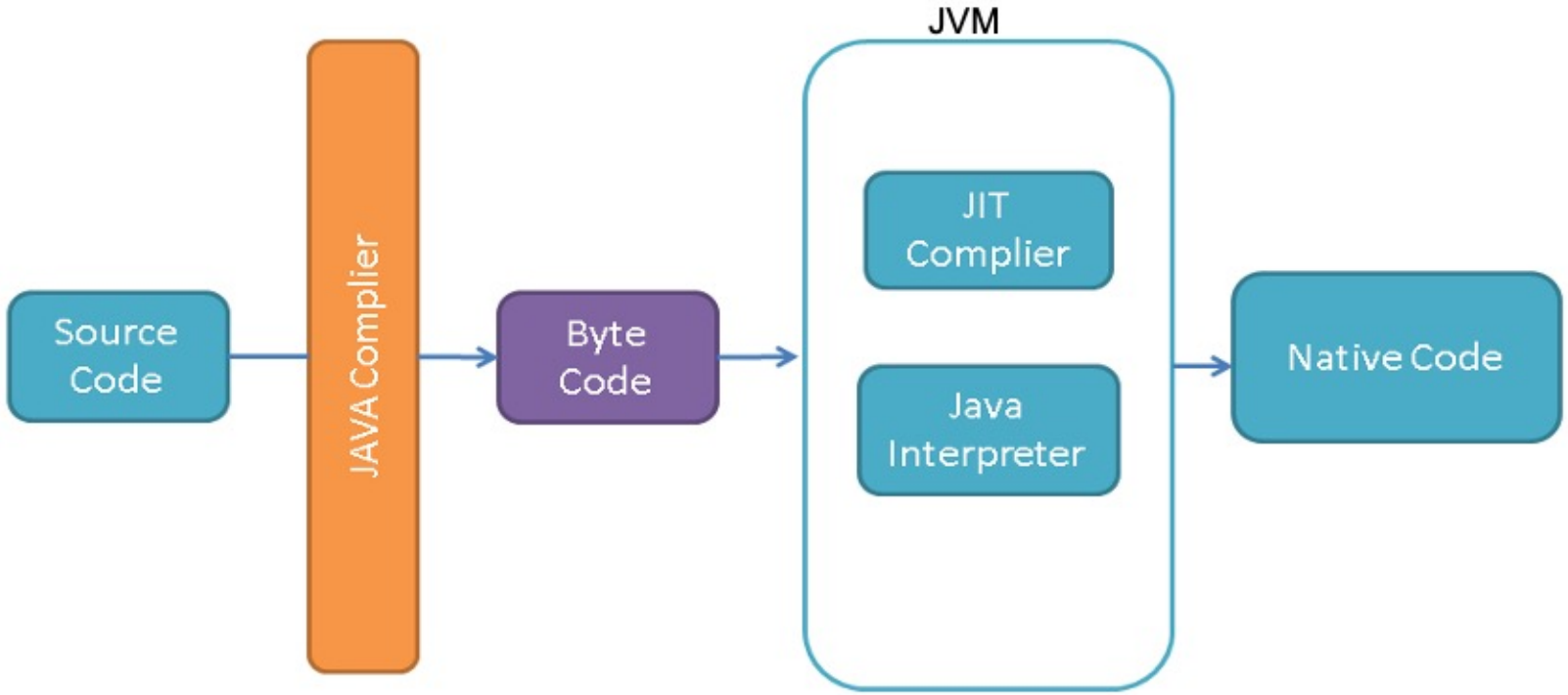
- javac este prescurtarea pentru compilatorul java.
- Este independent de platformă.
- Este responsabil pentru următoarele sarcini care sunt după cum urmează:
- a) Verifică eroarea sintaxială (eroare de sintaxă).
- b) Convertește codul sursă în cod de octeți (bytes) cu ajutorul mașinii virtuale Java (JVM).
- c) Adaugă, de asemenea, cod suplimentar la programul dvs., dacă este necesar.

KeyTraining

Java Interpreter

- Este dependent de platformă.
- Este responsabil pentru următoarele sarcini. Acestea sunt după cum urmează:
- a)Convertește codul de octeți (bytes) în cod nativ linie cu linie.
- b) Execută programul pe sistemul dvs.

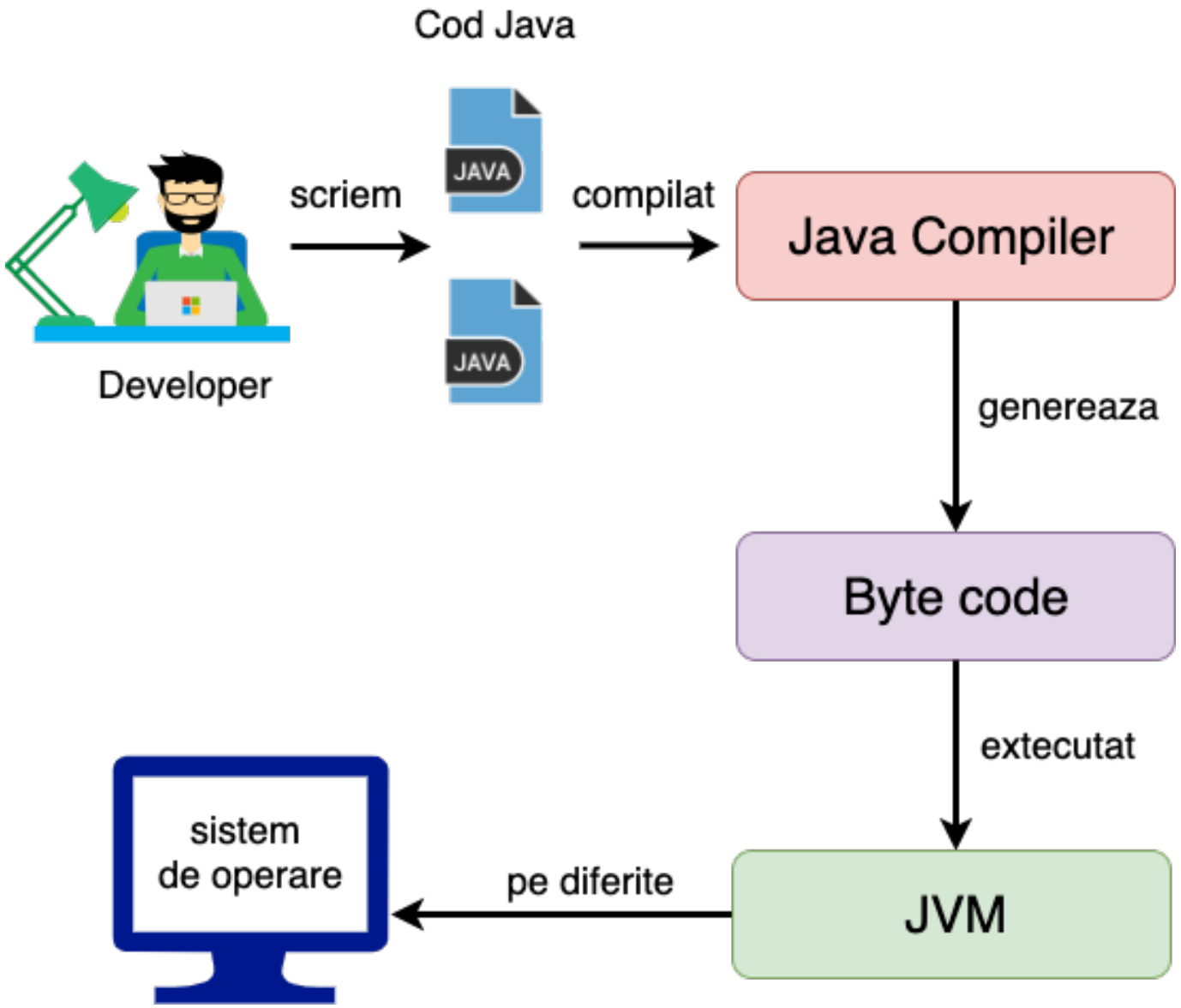
How does Java work?



Platform independent

Platform dependent

KeyTraining



KeyTraining

KeyTraining

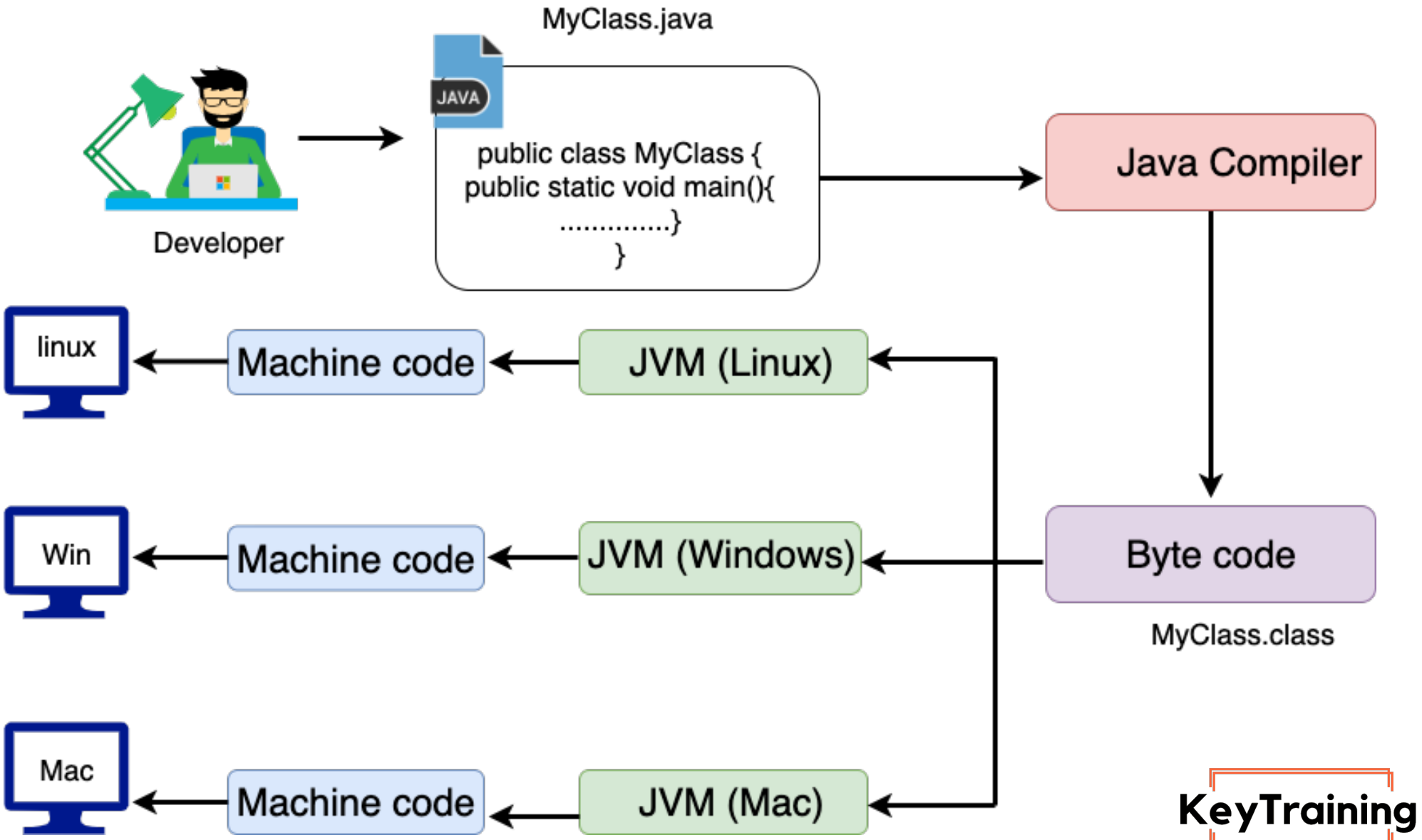
Bytecode in Java

- Bytecode în Java este un set extrem de optimizat de instrucțiuni pentru mașina virtuală Java (JVM) care le citește și interpretează pentru a rula programul java.
- Un bytecode este un cod de program binar care poate rula numai pe JVM.
- Cu alte cuvinte, este un limbaj mașină (cod) pentru JVM sub forma unui fișier .class, dar nu este specific mașinii, deoarece nu este un cod nativ.
- Bytecode acționează ca un limbaj intermediar independent de platformă (sistem).
- Este generat de un interpretor Java care poate fi rulat direct de o mașină reală.

KeyTraining

Bytecode in Java

- Bytecode-ul are două caracteristici importante care sunt după cum urmează:
- 1. Este independent de procesor, adică programul java poate fi executat pe orice arhitectură de procesor.
- 2. Nu depinde de sistemele de operare precum Windows, Linux și Mac OS.



KeyTraining

KeyTraining

Bytecode vs Machine code

- Principala diferență între machine code și bytecode, codul mașinii este un set de instrucțiuni în limbajul mașinii sau binar care poate fi executat direct de către CPU, in timp ce bytecode-ul este un cod generat prin compilarea unui cod sursă care nu poate fi rulat direct de către CPU si care se bazează pe un interpretor pentru a fi executat.

KeyTraining

JRE (Java runtime environment)

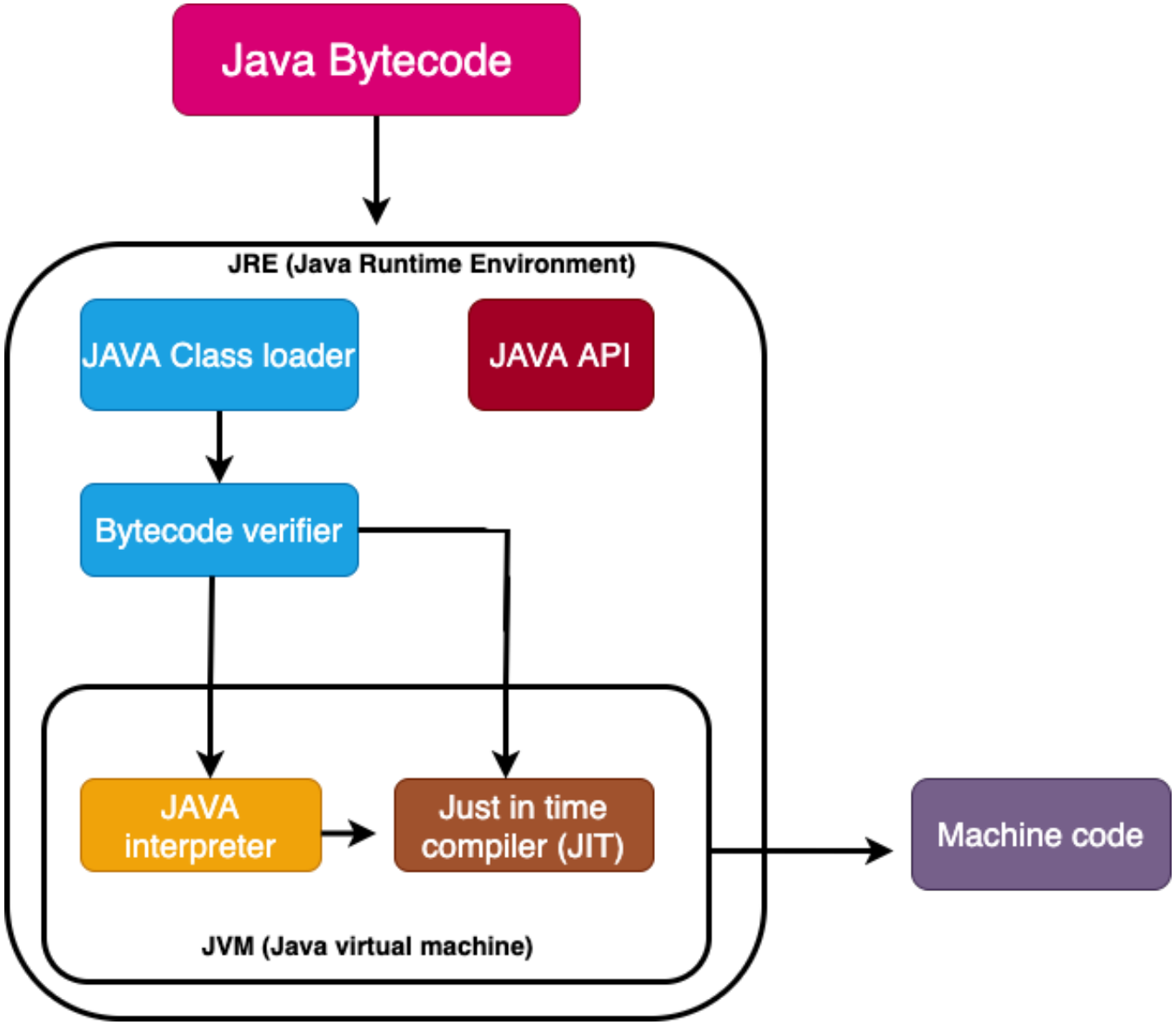
- JRE este un mediu software în care se execută programul Java.
- Cu alte cuvinte, JRE este un mediu software care este necesar pentru executarea aplicațiilor Java pe orice sistem, indiferent de sistemul de operare și hardware-ul de bază.
- JRE face parte din setul de dezvoltare Java, dar nu poate fi utilizat independent pentru a executa niciun bytecode (program java compilat).
- Nu poate fi utilizat pentru dezvoltarea programelor de aplicații Java, dar este utilizat pentru executarea acestora.
- Java Runtime Environment este o colecție de programe care conține mașină virtuală Java și librarii de clase care sunt necesare pentru a rula programe pe JVM (prin comanda java).

KeyTraining

JRE (Java runtime environment)

- Dacă sistemul computerului dvs. are instalat JDK (Java Development Kit) actualizat, nu este necesar să îl instalați separat.
- Java Development Kit conține, de asemenea, Java Runtime Environment (JRE) care permite rularea programului Java în orice IDE avem instalat.
- JRE constă din următoarele componente principale care sunt după cum urmează:

- Java API (Application Programming Interface)
- Class Loader
- Bytecode verifier
- Java Virtual Machine (Interpreter)



KeyTraining

KeyTraining

Java API

- Java (API) este o colecție foarte mare de componente software, gata făcute, care oferă funcționalitatea de bază a limbajului de programare Java.
- În cuvinte simple, Java API este o colecție mare de clase, interfețe și metode deja definite sub formă de pachete Java.
- Oferă numeroase capabilități utile, cum ar fi interfața grafică de utilizator (GUI), data, ora și capabilitățile calendarului pentru programatori.
- API-ul Java este grupat în librarii de clase și interfețe conexe, împreună cu câmpurile, constructorii și metodele lor.
- Aceste librarii sunt furnizate sub formă de pachete.

KeyTraining

Java API

- Oferă funcții de programare suplimentare construite pe platforma de bază Java.
- Înseamnă că caracteristicile de bază ale limbajului de programare Java nu se schimbă atunci când este lansată o nouă versiune a acestuia.
- Deoarece API-ul Java este flexibil, acesta poate fi deschis pentru a adăuga pachete sau librarii noi în acesta.
- Librarii populare și funcționalitatea lor din API-ul Java, pe scurt, sunt după cum urmează:

Java API

- Java.lang: Este un pachet care oferă clase fundamentale pentru proiectarea limbajului de programare Java. Pachetul Java.lang este importat dinamic (adică încărcat) într-un program Java. Nu este nevoie să importe în mod explicit. Cele mai utilizate clase din pachetul Java.lang sunt Double, Float, Integer, String, StringBuffer, System și Math.
- Java.io: Acceptă operatiuni de intrare / ieșire prin sistemul de fișiere, tastatură, rețea etc. Pachetul java.io conține mai multe clase pentru efectuarea operațiilor de intrare și ieșire. Cele mai utilizate clase din java.io sunt clasa File, InputStreams, OutputStreams, Readers, Writers și RandomAccessFile.

KeyTraining

Java API

- Java.util: acceptă diverse utilități de programare. Pachetul java.util oferă clase de colecție moștenite, event model, collections framework, capabilități de dată și oră și alte clase utilitare, cum ar fi string tokenizer.
- Java.math: Este folosit pentru a sprijini operațiile matematice.
- Java.security: acceptă funcții de securitate.
- Java.awt: acceptă crearea interfeței grafice de utilizator (GUI), pictarea graficelor și a imaginilor.
- Java.sql: suport pentru accesarea bazelor de date relaționale prin SQL.
- Java.beans: acceptă crearea de Java Beans.

KeyTraining

Java API

- Java.net: pachetul java.net oferă clase care sunt utilizate pentru implementarea rețelelor în programele java.
- Java.imageIO: Suportă pentru intrarea / ieșirea imaginii.
- Aceste API Java predefinite oferă o cantitate extraordinară de funcționalități de bază unui programator. Un programator ar trebui să fie conștient de aceste API-uri Java. El ar trebui să știe cum să utilizeze aceste API-uri Java.
- O listă completă a tuturor claselor în Java API poate fi găsită pe site-ul web Oracle: https://docs.oracle.com/javase/7/docs/api/
-

KeyTraining

Java class loader

- Când scriem un program în java, programul este plasat în memorie de către class loader înainte de a putea fi executat.
- Java Class Loader preia fișierul .class care conține bytecode și îl transferă în memorie.
- Încarcă fișierul .class de pe un disc de pe sistemul dvs. sau printr-o rețea. După încărcarea clasei, acesta este trecut la bytecode verifier.
- În principiu, există trei subcomponte ai class loader în Java:
  - Bootstrap class loader (incarca lib)
  - Extensions class loader (incarca ext)
  - System class loader (incarca tot din Classpath)

KeyTraining

Java bytecode verifier

- Bytecode verifier verifică dacă bytecode-urile sunt valide sau nu fără a încălca oricare dintre regulile de securitate Java.
- Oferă o atenție specială verificării tipului tuturor variabilelor și expresiilor din cod.
- Bytecode verifier se asigură că nu există acces neautorizat la memorie. Odată ce codul este verificat cu succes, acesta este transferat în Java Virtual Machine (JVM) pentru interpretare.

KeyTraining

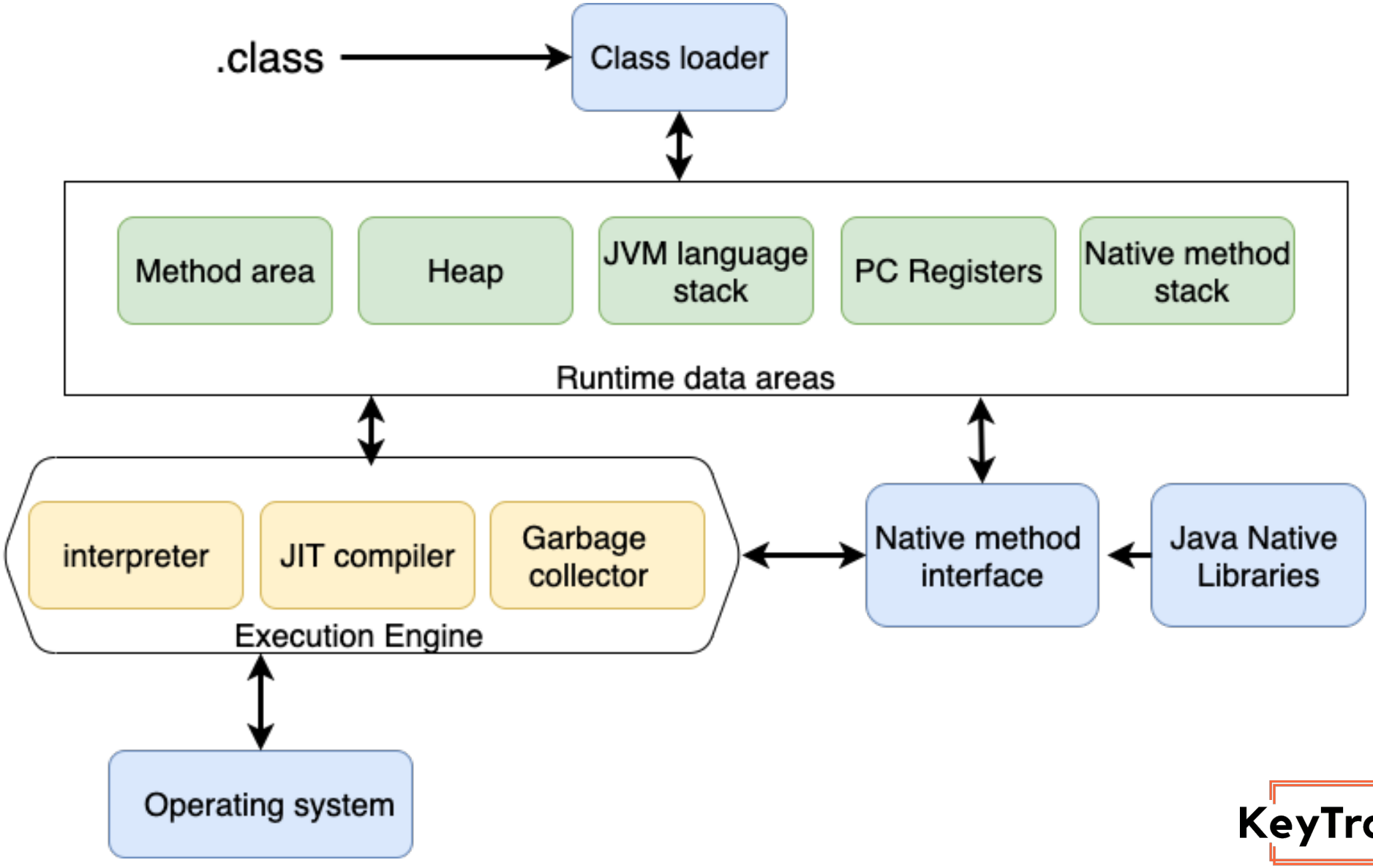
JVM - Java Virtual Machine

- Java Virtual Machine (JVM) în Java este inima întregului proces de execuție al programului Java. Este practic un program care oferă mediul de rulare necesar pentru executarea programelor Java.
- Cu alte cuvinte, Java Virtual Machine (JVM) este un computer abstract, care este responsabilă de executarea codului secundar Java (un set extrem de optimizat de instrucțiuni) pe o anumită platformă hardware.
- Se mai numește și sistem de rulare Java.
- Specificația JVM este furnizată de Sun Microsystem a cărei implementare oferă un mediu de rulare pentru a executa aplicațiile noastre Java.
- Implementarea JVM este cunoscută sub numele de Java Runtime Environment (JRE).

KeyTraining

JVM - Java Virtual Machine

- Deoarece JVM depinde de platformă, prin urmare, este disponibil pentru multe platforme hardware și software.
- Nu putem rula programul Java decât dacă JVM este disponibil pentru hardware-ul și platforma sistemului de operare corespunzător sistemul pe care rulam.



KeyTraining

KeyTraining

JVM - Java Virtual Machine

- Java Virtual Machine efectuează următoarele operațiuni pentru executarea programului. Acestea sunt după cum urmează:
- a) Încărca codul în memorie.
- b) Verifică codul.
- c) Execută codul
- d) Oferă mediu de rulare.
- Când realizăm un program în Java, codul java .java este convertit într-un fișier .class format din instrucțiuni de cod de bytes de către compilatorul Java. Acest compilator Java se află în afara JVM.

KeyTraining

JVM - Java Virtual Machine

- În JVM, class loader este un modul sau program care îndeplinește următoarele funcții:
- a) În primul rând, class loader încarcă fișierul .class în memorie.
- b) Apoi, bytecode verifier verifică dacă toate instrucțiunile bytecode sunt corecte sau nu. Dacă consideră că orice instrucțiune este suspectă, procesul de execuție ulterior este respins imediat.
- c) Dacă instrucțiunile de bytecode sunt corecte, aceasta alocă memoria necesară pentru a executa programul.
- Această memorie este împărțită în 5 părți separate care se numesc zone de date în timpul rulării.

KeyTraining

JVM - Java Virtual Machine

- Acesta zona conține datele și rezultatele în timpul executării programului. Aceste zone sunt după cum urmează:
- 1. Class (Method) Area: este un bloc de memorie care stochează codul clasei, codul variabilelor și metodele programului Java. Aici metode înseamnă funcții declarate în clasă.
- 2. Heap: Aceasta este zona de date de execuție în care sunt create obiecte. Când JVM încarcă o clasă, o metodă și o zonă de heap sunt construite imediat în ea.
- 3. Stacks : Codul metodei este stocat în Method area. Dar în timpul executării unei metode, este nevoie de mai multă memorie pentru a stoca datele și rezultatele. Această memorie este alocată pe Java stacks.
- Java stacks sunt acele zone de memorie în care sunt executate metodele Java. În Stack, se creează un cadru separat în care se execută metoda. De fiecare dată când se apelează o metodă, un nou cadru este creat în stacks. Când invocarea metodei este finalizată, un cadru asociat cu aceasta este distrus. JVM creează întotdeauna un thread separat (sau proces) pentru a executa fiecare metodă.

KeyTraining

JVM - Java Virtual Machine

- 4. PC Register : (Program Counter) sunt acele registre (zone de memorie) care conțin adresa de memorie a instrucțiunilor JVM în curs de executare.
- 5. Native Method Stack: Metodele programului Java sunt executate pe Java stack. În mod similar, metodele native utilizate în program sau aplicație sunt executate pe stive de metode native (Native method stacks.).
- În general, pentru a executa metodele native, sunt necesare librarii de metode native Java. Aceste fișiere antet sunt localizate și conectate la JVM printr-un program, cunoscut sub numele de Native method interface.

KeyTraining

JIT Compiler

- Compilatorul JIT în Java este partea JVM care este utilizată pentru a crește viteza de execuție a unui program Java.
- Cu alte cuvinte, este folosit pentru a îmbunătăți performanța de execuție a programului.
- Ajută la reducerea timpului necesar pentru executarea programului.

KeyTraining

Interpreter

- Interpreter în Java este un program de calculator (software) care implementează Java Virtual Machine (JVM) și rulează aplicații Java (programe).
- Traduce (convertește) bytecode în cod mașină (cod nativ) linie cu linie în timpul runtime-ului.
- Cu alte cuvinte, Interpreter in Java este un program care convertește programul de nivel înalt în limbaj de asamblare (limbaj mașină). Este folosit pentru executarea programelor Java. Interpretul ia bytecode ca intrare și execută acel cod convertindu-l în cod mașină.

KeyTraining

Interpreter

- Java Interpreter este responsabil pentru următoarele funcții care sunt după cum urmează:
- 1. Funcția principală a interpretului este de a converti instrucțiunea bytecode în codul mașină linie cu linie în timpul rulării, fără a schimba secvența.
- 2. Realizează toate activitățile sistemului Java runtime.
- 3. Rulează aplicații citind și interpretând fișiere bytecode.

KeyTraining

Interpreter

- Atunci când programul Java este compilat, compilatorul generează un cod intermediar (în formatul fișierului .class) cunoscut sub numele de bytecode pentru JVM.
- Java Virtual Machine există numai în memoria computerului.
- Nu este realizat din hardware sau alte componente fizice. Acest bytecode sunt independente de mașină și, prin urmare, pot fi executate pe orice mașină.
- Interpretorul Java citește bytecode linie cu linie și le convertește în cod de mașină pentru o mașină specifică pe care rulează programul Java.

KeyTraining

Interpreter

- Prin urmare, un Interpreter este scris individual pentru fiecare tip de mașină.
- JVM folosește interpretul la runtime și apoi execută codul pe mașina gazdă după ce a interacționat cu sistemul de operare.
- Multe browsere precum Google Chrome, etc. sunt browsere care conțin interpretorul Java.
- Aceasta înseamnă că acestea sunt browsere compatibile cu Java. Este folosit pentru a executa programul Applet în browser.

KeyTraining

```java
package demojava;

public class FirstJavaClass {

    public static void main(String[] args) {

        System.out.println("Primul meu program Java");

    }

}
```

Primul program Java

KeyTraining

Primul program Java

```
package demojava;

public class FirstJavaClass {

    public static void main(String[] args) {

        System.out.println("Primul meu program Java");

    }

}
```

- Această declarație declară o clasă numită FirstJavaClass.
- Aici, class este un cuvânt cheie în Java care este utilizat pentru a defini o clasă. FirstJavaClass este un identificator Java care reprezintă numele clasei de definit. Deoarece Java este un limbaj de programare orientat pe obiecte, totul trebuie plasat în interiorul unei clase.
- Fiecare definiție a clasei începe cu o acolada de deschidere „{” și se termină cu o acolada de închidere „{“.

KeyTraining

Primul program Java

```
package demojava;

public class FirstJavaClass {

    public static void main(String[] args) {

        System.out.println("Primul meu program Java");

    }

}
```

- Această afirmație definește metoda main ().
- Acesta este rulata automat la executarea programului.
- Fiecare program Java trebuie să aibă o metoda main (). Este punctul de plecare pentru ca interpretorul să înceapă procesul de execuție al programului java.
- Un program Java poate conține orice număr de clase, dar numai una dintre ele trebuie să conțină metoda main pentru a începe executarea.

KeyTraining

Primul program Java

- Această declarație conține un număr de cuvinte cheie, cum ar fi public, static, void si main
- a) public: cuvântul cheie public este un modificator de acces care reprezintă vizibilitate. Înseamnă că este accesibil tuturor celorlalte clase.
- b) static: este un cuvânt cheie. Dacă definim o metodă ca fiind statică, aceasta se numește metodă statică. Principalul avantaj al metodei statice este că nu este nevoie să creați un obiect pentru a apela metoda statică. Metoda principală trebuie declarată întotdeauna ca statică deoarece JVM execută metoda statică înainte de crearea obiectului. Înseamnă că JVM nu trebuie să creeze un obiect pentru a clasa metoda principală. Astfel, economisește și memorie.
- c) void: Este un modificator care afirmă că metoda principală nu returnează nicio valoare.
- d) main: Reprezintă punctul de plecare pentru execuția programului.
- e) String [] args: Conține o serie de obiecte de tipul clasei String. Este folosit pentru argumentele din linia de comandă.

KeyTraining

Primul program Java

```
package demojava;

public class FirstJavaClass {

    public static void main(String[] args) {

        System.out.println("Primul meu program Java");

    }

}
```

- Metoda println () este utilizată pentru a tipări (afișa) „Primul meu program Java” in consolă.
- Deoarece Java este un limbaj de programare pur orientat pe obiecte, fiecare metodă face parte dintr-un obiect.
- Metoda println () este o parte a obiectului out, adică un membru de date statice din clasa System.

KeyTraining

Primul program Java

```
package demojava;

public class FirstJavaClass {

    public static void main(String[] args) {

        System.out.println("Primul meu program Java");

    }

}
```

- Un program este compus dintr-un set de instrucțiuni care se numesc statements.
- Este necesar un punct și virgulă";" pentru a indica sfârșitul unui statement.
- Fiecare instrucțiune Java trebuie să se încheie cu punct și virgulă.

KeyTraining

Pachete in Java (packages)

- Un pachet Java reprezinta un grup de clase grupate in acelasi loc dupa un scop comun.
  - Pentru o mai buna organizare
  - Pentru rezolvarea conflictelor ce tin de nume
  - Pentru limitarea accesului (acces modifiers)

Pachete in Java (packages)

- Prima linie a unui program Java va fi intodeauna declaratia legata de apartenenta clasei fata de un pachet.

```
package curs1;

public class BugsCalculator {

    ///code here
}
```

Pachete in Java (packages)

- Daca o clasa nu este creata in interioul unui pachet, atunci ea este atribuita pachetului default si astfel declaratia package nu mai exista.

```
public class TestClass {

}
```

Pachete in Java (packages)

- Denumirea pachetelor :
  - Folosind numai litere mici (poate contine cifre)
  - Este recomandat ca firmele care dezvolta software sa foloseasca in numele pachetelor DNS-ul (domain name service) lor plecand de la extensie.
  - Exemplu : companie.com => com.companie.utils
  - => com.companie.gui

Clase si obiecte

- O clasa reprezinta un template/sablon pentru obiecte.
- Pe baza clasei construim obiecte.
- Continutul si modul de manifestare al obiectului este definit in clasa prin atribute si metode.
- Astfel un program Java este un ansamblu de obiecte care interactioneaza intre ele, unde fiecare dintre aceste obiecte reprezinta o functionalitate din scopul programului.

Clasa Tester

```
public class Tester {

    String name;
    int age;
    int ratePerHour;

    public Tester(){

    }

    public void runTests() {

    }

    public void writeAutomationScripts() {

    }
    public void insertBugs() {

    }
```

Clasa Tester

```
public class Tester {

    String name;
    int age;
    int ratePerHour;

    public Tester(){

    }

    public void runTests() {

    }

    public void writeAutomationScripts() {

    }
    public void insertBugs() {

    }
```

state/atributes

Clasa Tester

```
public class Tester {

    String name;
    int age;
    int ratePerHour;

    public Tester(){

    }

    public void runTests() {

    }

    public void writeAutomationScripts() {

    }
    public void insertBugs() {

    }
```

state/atributes

constructor

Clasa Tester

```
public class Tester {

    String name;
    int age;
    int ratePerHour;

    public Tester(){

    }

    public void runTests() {

    }

    public void writeAutomationScripts() {

    }
    public void insertBugs() {

    }
```

state/atributes

constructor

behaviour/methods

Clase si obiecte

- Pentru a crea un obiect dintr-o clasa folosim cuvantul NEW

```
Tester tester1 = new Tester();
```

- Crearea de obiecte dintr-o clasa se numeste Instantiere
- Fiecare obiect este o instanta a clasei
- Putem avea oricate obiecte vrem dintr-o clasa

Clase si obiecte

- Caracteristicile unui obiect in Java
- **1. State**: State reprezintă proprietățile unui obiect. Este reprezentat de variabila de instanță / atributul unui obiect. Proprietățile unui obiect sunt importante, deoarece rezultatul funcțiilor depinde de proprietăți.
- **2. Comportament**: Comportamentul reprezintă funcționalitate sau acțiuni. Este reprezentat de metode în Java.
- **3. Identitate**: Identitatea reprezintă numele unic al unui obiect. Diferențiază un obiect de celălalt. Numele unic al unui obiect este utilizat pentru a identifica obiectul.

```
/**
 * Definim Tester1
 */
Tester tester1 = new Tester();
tester1.setName("Matei");
tester1.setRatePerHour(20);
/**
 * Definim Tester2
 */
Tester tester2 = new Tester();
tester2.setName("Oana");
tester2.setRatePerHour(25);
```

Obiectul Tester

Nume obiect

```java
public class Tester {

    String name;
    int age;
    int ratePerHour;

    public Tester(){

    }

    public void runTests() {

    }

    public void writeAutomationScripts() {

    }
    public void insertBugs() {

    }
```

```java
/**
 * Definim Tester1
 */
Tester tester1 = new Tester();
tester1.setName("Matei");
tester1.setRatePerHour(20);
/**
 * Definim Tester2
 */
Tester tester2 = new Tester();
tester2.setName("Oana");
tester2.setRatePerHour(25);

tester1.insertBugs();
tester2.insertBugs();
```

Clase si obiecte

- Diferente intre clasa si obiect:
- 1. O clasă este un tip de date definit de utilizator, în timp ce un obiect este o instanță a tipului de date de clasă.
- 2. O clasă generează obiecte în timp ce un obiect dă viață unei clase.
- 3. Clasele nu ocupă locația memoriei, dar obiectele ocupă locația memoriei.
- 4. Clasele nu pot fi manipulate deoarece nu sunt disponibile în locația memoriei, dar obiectele pot fi manipulate.

Structura unei clase

- Acces modifier
- Class name
- Body

```
public class Tester {

    String name;
    int age;
    int ratePerHour;

    public Tester(){

    }

    public void runTests() {

    }

    public void writeAutomationScripts() {

    }
    public void insertBugs() {

    }
```

Structura unei clase

- **Modifiers** : O clasa poate avea un modificator de acces Public sau Default. Membrii clasei pot fi : Public, Private, Protected sau default.

- **Name:** Prin convenție, un nume de clasă ar trebui să înceapă cu o literă mare și cu caractere ulterioare cu litere mici (de exemplu Student).
- Dacă un nume constă din mai multe cuvinte, prima literă a fiecărui cuvânt ar trebui să fie cu majuscule (de exemplu TesterManual).
- Deasemenea, exista asa numitele cuvinte cheie/rezervate (reserved word) care nu pot fi folosite.

Structura unei clase

- **Body** : Continutul unei clase sta intre o perecehe de {}
- **O clasa poate contine urmatoarele elemente :**
- Fields: Câmpurile sunt variabile membre ale unei clase care stochează date sau valori în ea. Specifică starea sau proprietățile clasei și obiectul acesteia. O variabilă poate fi o variabilă locală, variabilă de instanță sau variabilă statică.
- Constructor: Un constructor este utilizat pentru a crea un obiect. Fiecare clasă trebuie să conțină cel puțin un constructor în caz contrar, niciun obiect nu poate fi creat din clasă. Daca nu specificam un constructor, se aloca unul default.

Structura unei clase

- **Body** : Continutul unei clase sta intre o perecehe de {}
- **O clasa poate contine urmatoarele elemente :**
- Methods: o metodă definește acțiunea sau comportamentul clasei pe care o poate efectua obiectul unei clase. Are un corp în interiorul parantezelor. În corp, scriem cod care efectuează acțiuni. Poate fi o metodă de instanță sau o metodă statică.
- Main method: O clasă are și metoda principală care oferă punctul de intrare pentru a începe executarea oricărui program.