Laurențiu Spilcă

Java

Ghidul practic și interactiv pentru începători

București
2024

Cuprins

1

Java - Ghidul practic și interactiv pentru începători

Cum să începi simplu

În acest capitol

- Ce este și de ce să înveți Java
- Scrie primul tău program în Java
- Ce vei învăța în această carte

Dacă ai deschis această carte, sunt destul de sigur că ai o oarecare idee despre ce este Java și ai decis să înveți mai multe despre acest limbaj de programare. Java este unul dintre cele mai populare și utilizate limbaje de programare în zilele noastre, iar interesul tău este să dobândești abilitățile necesare pentru a dezvolta programe folosind acest limbaj. Ești la început de drum în software? Sau mai ai experiență cu alte limbaje de programare? Oricare ar fi situația în care te afli, această carte te va ajuta cu siguranță să înțelegi mai bine cum se folosește Java într-un mod profesional.

Te voi lua de la zero – sau altfel spus, voi considera că nu ai văzut până acum nicio linie de cod scrisă cu Java sau un alt limbaj (deși trebuie să recunosc că ai avea un avantaj substanțial dacă ai mai lucrat cu oricare alt limbaj de programare înainte, în facultate, în liceu sau un anterior loc de muncă). Însă spre deosebire de alte cărți, te voi ghida pe un drum practic. Voi evita să-ți dau detalii care nu sunt strict necesare, pentru început, în cariera ta, și te voi învăța exact ceea ce l-aș învăța pe un începător care ajunge în echipa mea. Dacă totuși partea de început a cărții ți se pare cunoscută, poți continua direct cu unul din capitolele care prezintă mai mult interes pentru tine. Dar chiar și așa, poate te va ajuta să le citești și pe primele, măcar la suprafață, pentru a-ți aminti detalii pe care poate le-ai uitat.

Așa cum ți-am promis, în acest prim capitol începem să discutăm noțiunile elementare. Întâi îți voi spune ce este Java și cum se folosește acest limbaj de programare din perspectiva unei persoane cu experiență. Pentru că vreau să fac această carte cât mai practică și (sper!) să nu te plictisesc, vom scrie deja un prim program care ne va ajuta să verificăm că ai instalat tot ceea ce ne va trebui mai departe în exemplele din

această carte. Toate exemplele din această carte sunt disponibile la adresa următoare: **https://link.telacad.ro/java/incepatori**.

Cu toate că vei avea acces la proiectele gata rezolvate, eu te sfătuiesc să lucrezi împreună cu mine aceste exemple din carte. Foloseşte exemplele rezolvate doar atunci când eşti blocat (nu înţelegi de ce nu funcţionează ce ai făcut şi eşti în impas), sau la final pentru a verifica ceea ce ai lucrat singur.

Haide! Să începem această călătorie despre cum să abordezi practic scrierea programelor cu Java.

1.1. Ce este şi de ce să învăţ Java

În această secţiune aş vrea să mă asigur că ştii ce este Java şi de ce merită să înveţi să lucrezi cu acest limbaj. Poate eşti deja la curent cu faptul că Java este foarte popular. Să discutăm cele mai relevante aspecte cu privire la acest limbaj şi motivele solide pentru care chiar ai vrea să începi această călătorie de studiu.

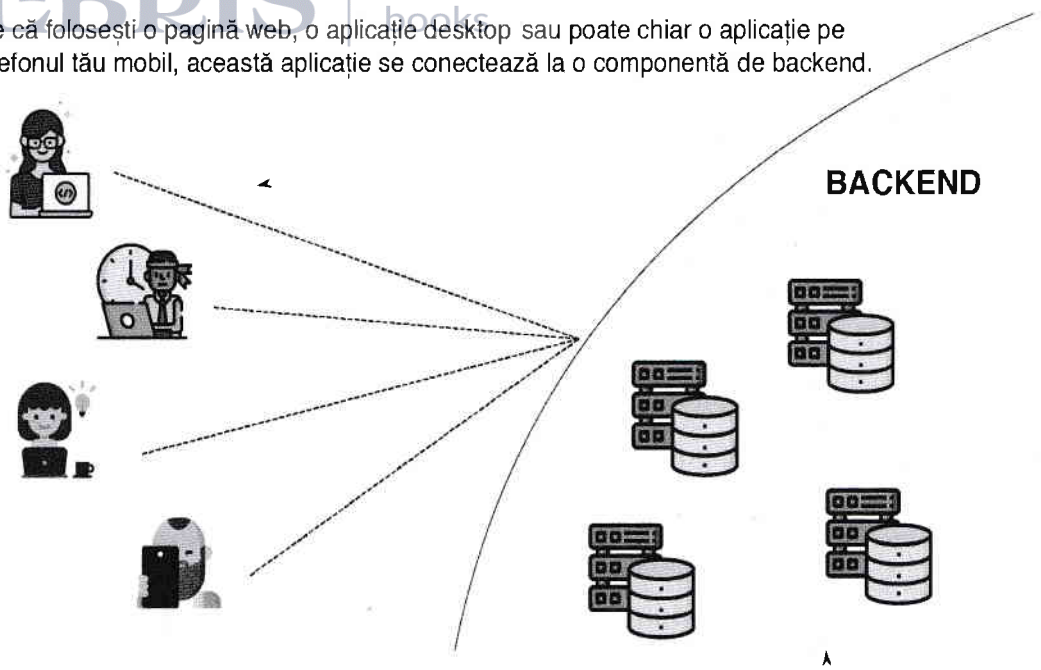
În anul 2021, conform unui sondaj realizat de Statista **(https://link.telacad.ro/java/cap1-1)**, Java este pe locul 5 în topul limbajelor de programare, dar în acelaşi timp locul 1 ca limbaj de programare preferat pentru dezvoltarea aplicaţiilor de tip backend.

Aplicaţiile de tip backend sunt instalate de obicei pe servere şi implementează logica de procesare şi păstrare a datelor pentru sisteme complexe. Orice aplicaţie pe care o foloseşti de pe un dispozitiv mobil (de exemplu, Android sau iOS) sau dintr-un browser are o aşa numită componentă de backend – o aplicaţie undeva ascunsă ochilor utilizatorilor dar care se ocupă de majoritatea acţiunilor de procesare a datelor cerute în urma evenimentelor generate de utilizatori (şi nu numai).

În acelaşi timp, spunem că orice aplicaţie care interacţionează direct cu utilizatorul este o componentă de frontend. În cele mai multe cazuri astăzi, componentele de frontend sunt aplicaţii web ce se execută prin intermediul unui browser. Aplicaţia ce se execută în browser când tu deschizi o pagină web este o aplicaţie de frontend (cel mai probabil scrisă folosind limbajul JavaScript şi tehnologii adiacente acestuia). Componentele de frontend comunică făcând schimb de date cu cele de backend pentru a realiza comportamentul final al sistemului.

Java este un limbaj extrem de popular pentru dezvoltarea acestor componente de tip backend pentru sisteme complexe. Când spun sisteme complexe, mă refer la sisteme care procesează o cantitate mare de date şi un număr larg de evenimente. Imaginează-ţi sistemul compus din toate aplicaţiile folosite de o bancă (atât cele necesare angajaţilor cât şi cele folosite de utilizatorii băncii, precum aplicaţiile de plăţi de pe telefonul mobil, din browser sau ATM-uri). Acestea sunt sisteme în care, de cele mai multe ori, componentele de backend sunt dezvoltate cu Java.

Fie că folosești o pagină web, o aplicație desktop sau poate chiar o aplicație pe telefonul tău mobil, această aplicație se conectează la o componentă de backend.



Componentele de backend ale unei aplicații se ocupă cu procesarea și memorarea datelor aplicațiilor client (pe care le folosește utilizatorul).

Figura 1.1 Componentele de backend ale unei aplicații sunt invizibile utilizatorilor. Cu toate acestea, componentele de backend implementează majoritatea operațiilor de procesare și stocare a datelor unei aplicații.

Sistemele bancare nu sunt, desigur, singurul exemplu. Majoritatea sistemelor de aplicații largi au componente scrise cu Java. Oricând auzi de un sistem de largă anvergură poți băga mâna în foc că există pe acolo măcar câteva aplicații (componente ale sistemului) scrise cu Java. Rețele de socializare (Facebook, Instagram, Twitter etc.), aplicații de transport (Uber, Bolt, Lyft etc.), aplicații pentru livrări (Uber Eats, Bolt, Glovo etc.). Toate acestea sunt exemple de aplicații larg utilizate, care, undeva în spatele lor, ascund, probabil, cel puțin câteva componente de backend scrise cu Java.

Concluzia este că, învățând Java, nu ai cum să nu îți găsești un loc de unde să scrii câteva linii de cod pentru un salariu mai mult decât rezonabil. Java este, de asemenea, un limbaj simplu, care îți deschide multe porți spre alte tehnologii pe care vei dori să le înveți mai departe. Și nu în ultimul rând, Java este un limbaj de programare care va fi cu siguranță folosit decade de acum înainte.

Dacă întrebarea ta încă mai este: Oare este ok să învăț Java? Răspunsul meu este unul foarte clar: DA. Nu ai decât de câștigat învățând Java!

1.2. Cum se foloseşte Java în practică, de fapt?

Dar cum se foloseşte Java în practică, de fapt? Dacă ai trecut deja prin secţiunea 1.1, ştii că Java este în cea mai mare măsura folosit astăzi pentru dezvoltarea componentelor de tip backend – acele aplicaţii care fac parte dintr-un sistem complex, nu sunt direct vizibile utilizatorilor, dar se ocupă de tot ce înseamnă procesarea şi stocarea datelor cu care sistemul lucrează. În această secţiune voi pune accentul pe ceea ce eu numesc obiectivul unui limbaj de programare. Care este obiectivul unui limbaj de programare şi de ce este important să înţelegi care este acesta pentru limbajul pe care vrei să-l înveţi? Obiectivul unui limbaj de programare îţi spune pentru ce fel de aplicaţii este gândit un anumit limbaj de programare.

Orice limbaj de programare are avantaje şi dezavantaje. Avantajele şi dezavantajele unui limbaj îl fac să fie destinat mai degrabă unui anumit tip de aplicaţii decât altul. De exemplu, aşa cum vom discuta mai în detaliu în secţiunea 1.5, execuţia tuturor aplicaţiilor Java se bazează pe un emulator (numit şi maşină virtuală, sau pe scurt JVM [Java Virtual Machine]). Acest aspect face ca aplicaţiile Java să fie mai uşor de dezvoltat, însă mai puţin performante (nu fac la fel de eficient managementul procesării şi uzul memoriei precum aplicaţiile scrise în alte limbaje). Desigur, îţi voi explica în secţiunea 1.5 de ce se întâmplă acest lucru. Însă, ca exemplu, este un motiv bun pentru care nu multe jocuri video sunt dezvoltate cu Java (da, Minecraft este o excepţie). Un limbaj precum C, sau C++ are alte caracteristici (avantaje şi dezavantaje). Aplicaţiile scrise cu C şi C++ au execuţia mult mai performantă decât cele scrise cu Java, însă, sunt mult mai dificil de scris, iar acest lucru face implementarea acestor aplicaţii mai costisitoare.

Aruncă un ochi pe tabelul 1.1 pentru a-ţi face o idee despre obiectivele principalelor limbaje de programare.

1.3. Primul lucru pe care trebuie să îl faci

În această secţiune vom începe pregătirile pentru a scrie programe. Nu sunt multe lucruri pe care trebuie să le faci înainte de a putea să scrii primul tău program cu Java. Ai de făcut, de fapt, doar doi paşi:

1. Să instalezi un mediu de implementare de aplicaţii, cunoscut şi ca IDE (Integrated Development Environment);

2. Să instalezi pachetul Java pentru dezvoltatori, cunoscut ca JDK (Java Developer Kit).

De fapt, astăzi, primul pas, în cele mai multe cazuri, îl include şi pe al doilea. Noi vom folosi un mediu de dezvoltare numit IntelliJ IDEA. Versiunea Community, pe care o poţi descărca de pe site-ul oficial **(https://link.telacad.ro/java/cap1-2)** şi o poţi folosi gratuit este suficient pentru ce ai nevoie ca să implementezi exemplele din această carte. De fapt, IntelliJ IDEA Community este des folosit şi în dezvoltarea aplicaţiilor reale. Deşi multe companii preferă achiziţionarea unei licenţe Ultimate, care adaugă capabilităţi ce fac munca programatorului mai confortabilă, aceste

capabilități nu sunt esențiale în dezvoltarea programelor. Din acest motiv, vei întâlni multe firme mai mici care nu investesc în aceste licențe.

IntelliJ IDEA nu este singurul IDE care există. Alte două IDE-uri apreciate și folosite în practică sunt Eclipse și NetBeans. Pe lângă acestea, unii dezvoltatori folosesc JDeveloper sau chiar și Visual Studio (care este mai degrabă cunoscut pentru limbaje din familia .NET sau JavaScript).

Limbaj de Program	Obiectiv
Java	• Componente tip backend • Aplicații de testare automată • Mai rar, componente tip desktop (care rulează direct pe calculatorul utilizatorului)
Python, R	• Machine learning • Algoritmi matematici • Mai rar, implementarea componentelor de tip backend
C, C++	• Aplicații care trebuie să aibă o execuție rapidă (de exemplu: jocuri video) • Mai rar, aplicații de tip backend sau desktop
C#	• Componente tip backend • Mai rar componente tip desktop sau jocuri video
JavaScript	• Aplicații de tip frontend (care rulează in browser) • Mai rar și componente de tip backend

Tabelul 1.1 Obiectivele principale în funcție de limbajul de programare.

Notă! Ceea ce vei învăța despre Java în această carte nu este dependent de folosirea unui anumit IDE. Eu a trebuit să aleg un IDE cu care să prezint exemplele, și am ales IntelliJ IDEA pentru că, statistic, este cel mai folosit IDE în momentul de față de programatorii Java. Tu însă, poți folosi un alt IDE la alegere. De fapt, eu chiar îți recomand să încerci mai multe tipuri și să nu te acomodezi numai cu unul.

Dacă vrei să urmărești îndeaproape exemplele din această carte, atunci instalează IntelliJ IDEA Community de aici: **https://www.jetbrains.com/idea/ download/.** Aplicația se instalează ca orice program, conform sistemului tău de operare, și nu necesită nicio configurare specială. Odată instalat, îl vei putea deschide pentru a crea un nou proiect. Figura 1.2 îți arată cum să creezi un proiect nou, folosind butonul New Project din fereastra IDE-ului.

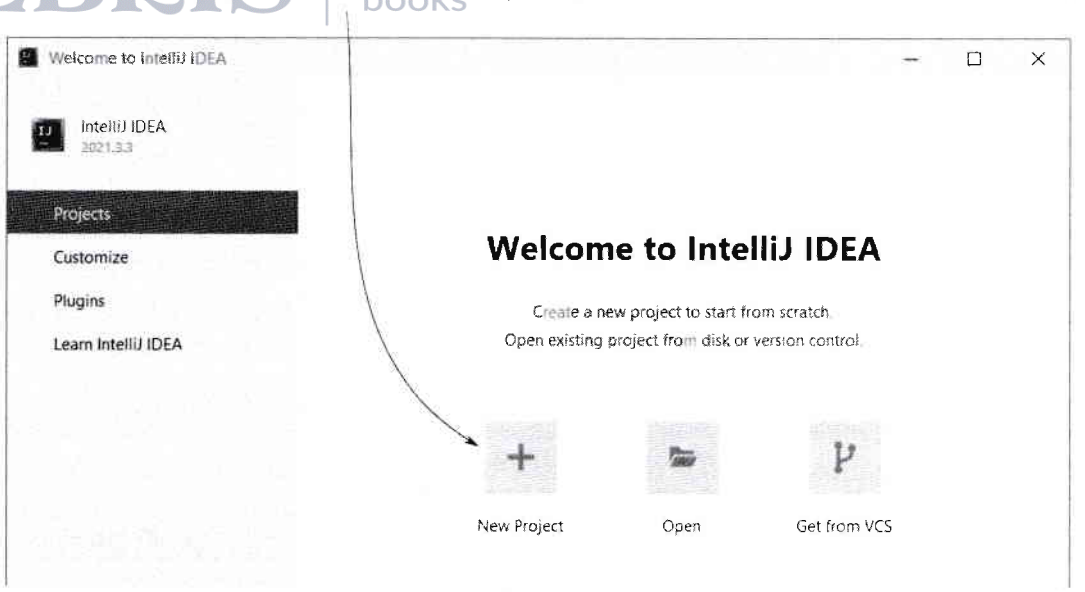


Figura 1.2 După ce ai instalat IntelliJ IDEA Community, poți crea un proiect nou folosind butonul New Project.

În secțiunea 1.4, vom crea un proiect nou și vom scrie primul program cu Java.

1.4. Scrie primul tău program în Java

Dacă acum ai tot ce este nevoie instalat pe calculatorul tău, putem începe scrierea primului program. În această secțiune vei învăța să creezi un proiect, să scrii și să execuți primul program. Aplicația pe care o vom face va fi una simplă – un program care afișează un mesaj sub forma unui text, de exemplu „Salutare prieteni!". Îți voi arăta cum se fac toate acestea pas cu pas. Desigur, voi presupune că ai instalat deja IDE-ul tău. În cazul în care nu ai făcut încă acest lucru, ar fi mai bine dacă ai începe cu secțiunea 1.3 unde am discutat de ce un IDE este necesar și cum să instalezi unul.

Haide să creăm un proiect nou folosind IDE-ul IntelliJ Community. După ce ai selectat butonul New Project, va trebui să alegi tipul proiectului tău. Alege Java (așa cum este prezentat în figura 1.3) ca tip al proiectului, apoi versiunea de Java pe care vrei să o folosești. Pentru exemplele din această carte vom folosi Java 21. Apasă pe butonul Next pentru a trece mai departe în configurarea proiectului.

În cazul în care nu găsești opțiunea pentru a selecta versiunea de Java pe care dorești să o folosești, va trebui să o descarci în sistemul tău. Acest lucru se face foarte simplu, selectând opțiunea **Download JDK**, așa cum observi în figura 1.4.

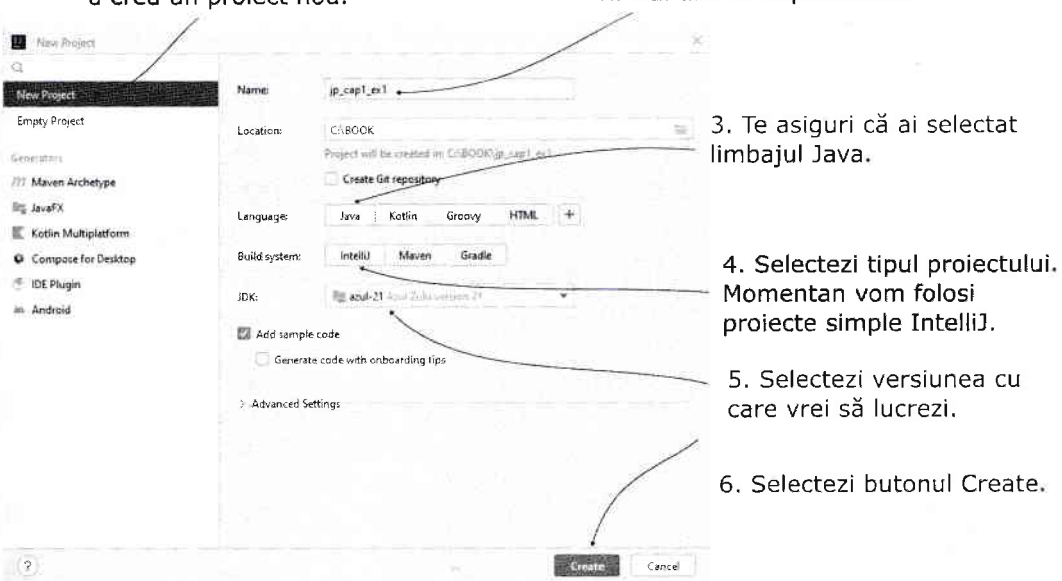


Figura 1.3 Pentru a crea un nou proiect, selectează tipul proiectului – Java, și versiunea pe care vrei să o folosești. Pentru exemplele din această carte, vom folosi Java 21, prin urmare, și tu trebuie să selectezi versiunea 21.

Dacă nu ai opțiunea de a selecta versiunea pe care dorești să o folosești, poți să o descarci folosind butonul de **Download JDK**

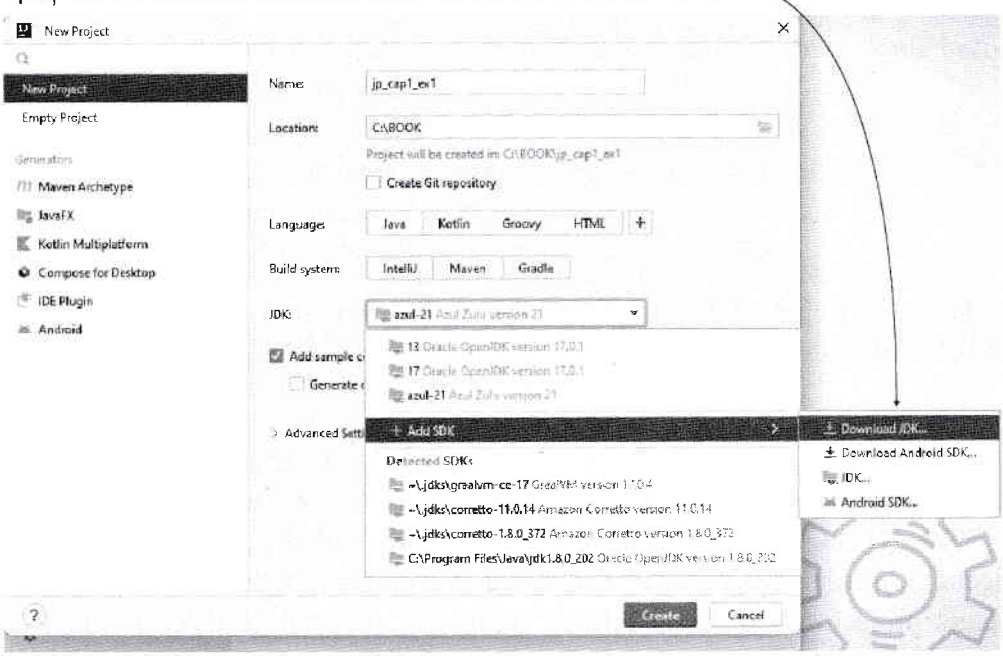


Figura 1.4 Poți descărca foarte ușor o versiune de JDK folosind direct butonul Download JDK din IDE. Ulterior descărcării, vei putea folosi această versiune a limbajului pentru a crea proiecte noi.

Figura 1.5 îți arată ce trebuie să faci după ce apeși pe Download JDK. Cel mai important lucru este să selectezi versiunea de Java pe care dorești să o descarci. De exemplu, în figura 1.5, eu aleg să descarc Java 21. Opțional, poți alege o anumită distribuție de JDK dar și o cale personalizată unde acesta se va descărca. Distribuțiile sunt implementări diferite puse la dispoziție de vendori diferiți. Din fericire, distribuția pe care o alegi nu va influența în niciun fel ceea ce înveți în această carte, și, în general, acesta va fi foarte rar un subiect de interes într-o aplicație reală pentru un programator. Din acest motiv mă voi limita la a mai da detalii cu privire la acest aspect. Dacă ești totuși interesat să afli mai multe despre distribuții Java, poți începe cu următorul articol: **https://link.telacad.ro/java/cap1-3.**

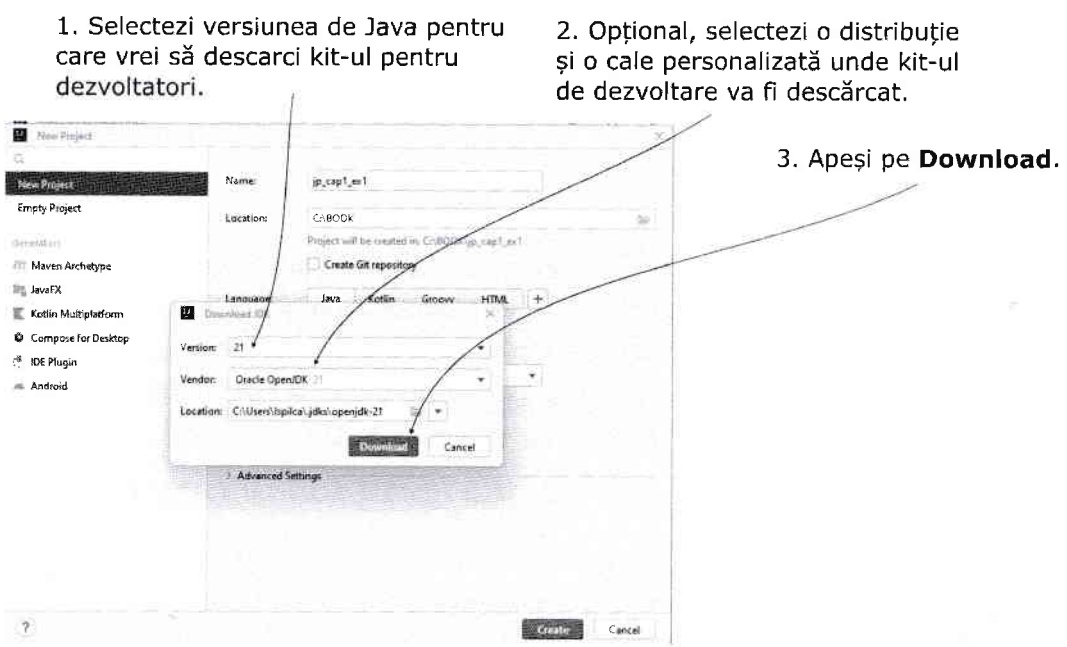


Figura 1.5 Alege versiunea de Java pe care vrei să o descarci pentru a dezvolta programe. Opțional, poți selecta o anumită distribuție și o cale personalizată unde aceasta să fie descărcată în sistemul tău.

La final, va trebui desigur să dai un nume proiectului tău și poți alege unde să fie acesta stocat în sistemul de fișiere al calculatorului tău. Odată ce ai făcut acest lucru, apasă butonul **Create** pentru ca IDE-ul tău să finalizeze crearea proiectului.

După câteva secunde de care IDE-ul are nevoie pentru a finaliza crearea proiectului, găsești structura acestuia ca cea prezentată în figura 1.6. IDE-ul prezintă structura proiectului (așa cum este ea definită în sistemul de fișiere) în partea din stânga a ferestrei. Pentru noi, deocamdată cel mai important dintre folderele prezente în structura proiectului este folderul „src" (prescurtare de la „source code" din engleză, „cod sursă"). Acest folder este cel în care vom stoca fișierele ce compun programul pe care îl scriem.

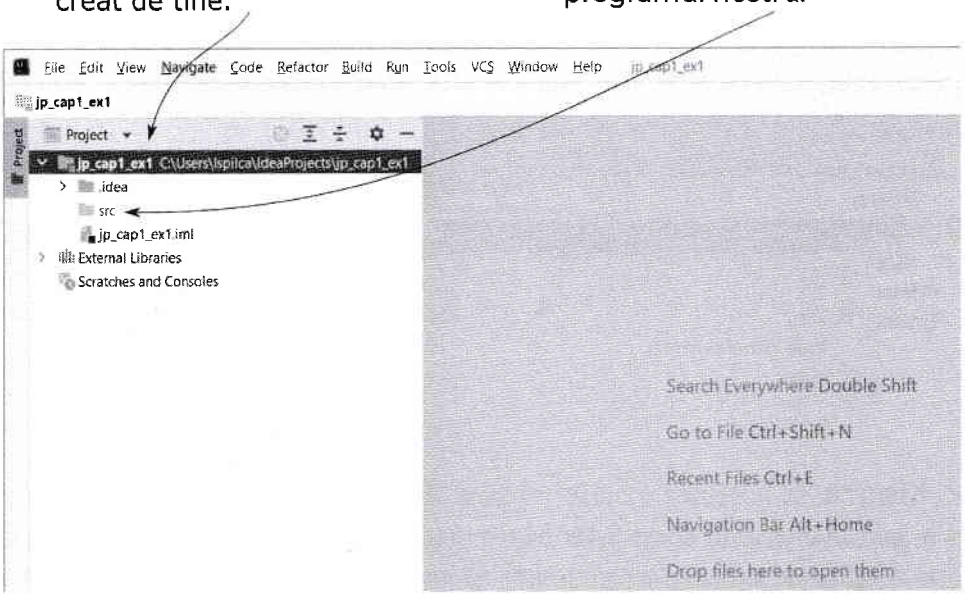


Figura 1.6 Structura proiectului. Folderul „src" este cel în care vom crea fișierele ce compun programul pe care îl scriem.

Acum că avem structura proiectului, putem să scriem primele linii de cod pentru ca programul nostru să facă ceva. În Java, logica programului se scrie în fișiere cu extensia .java. Aceste fișiere definesc clase.

Vom discuta mai în detaliu despre clase în capitolul 2, însă, pentru început am să te rog să faci abstracție de această noțiune. Deocamdată este important să înțelegi doar că vom crea un fișier cu extensia .java în interiorul folderului „src" al proiectului, iar în acest fișier vom scrie codul care definește programul.

Notă! Întotdeauna când dezvolti o aplicație cu Java vei scrie codul sursă în fișiere cu extensia .java.

Figura 1.7 îți arată cum să creezi un fișier cu extensia .java în folderul „src". Vom denumi acest fișier **Main** (în engleză, „principal") pentru că acesta reprezintă principalul (și, desigur, în cazul nostru singurul) fișier al proiectului.

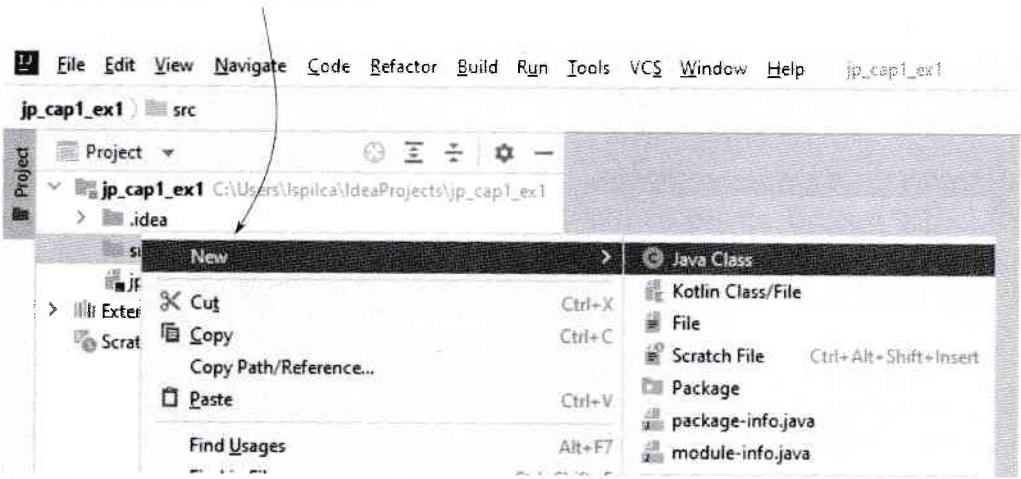


Figura 1.7 Pentru a scrie primele linii de cod, vom crea un fișier cu extensia .java. Vom numi acest fișier **Main** pentru că el este fișierul principal care definește programul nostru. Pentru a crea acest fișier, dă click dreapta pe folderul „src” din structura proiectului, apoi selectează opțiunea Java Class.

În fereastra ce apare, scrie numele fișierului pe care îl creezi. Este suficient să scrii numele, extensia .java fiind automat atribuită de IDE. După ce ai scris numele, execută dublu click pe opțiunea Class (figura 1.8).

În figura 1.9 poți vedea cum este prezentat fișierul nou creat Main.java de către IDE. Conținutul acestui fișier pare în partea dreaptă a structurii proiectului. IDE-ul adaugă câteva linii de cod predefinite în fișier. Aceste linii de cod, așa cum vei învăța în capitolul 2, reprezintă definiția clasei.

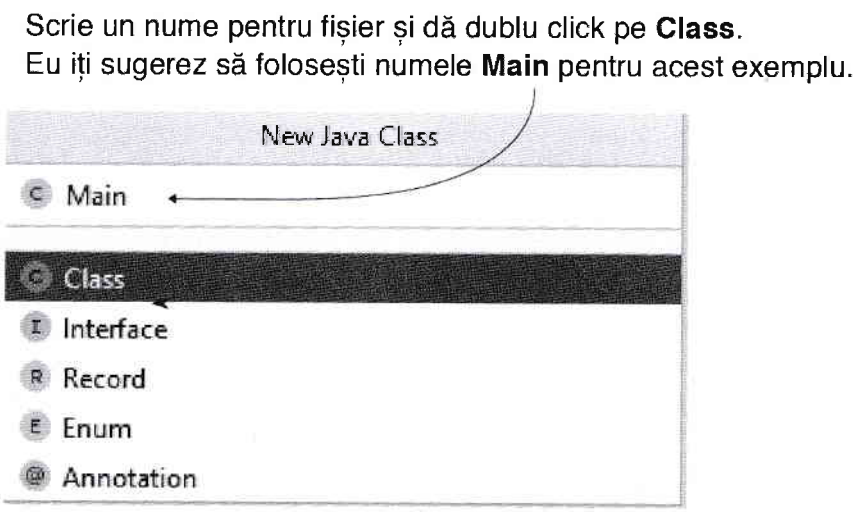


Figura 1.8 Alege numele fișierului de cod sursă și selectează opțiunea Class pentru a crea acest fișier.