

LBRIS

We know
books

Laurențiu Spilcă

Java

*Ghidul practic și
interactiv pentru
începători*

București

2024

Cuprins

Cuvânt înainte	15
Telecom Academy	17
Prefață	19
Mulțumiri	21
Despre autor	23
Partea I - Introducere	25
1. Cum să începi simplu	27
1.1. <i>Ce este și de ce să înveți Java</i>	28
1.2. <i>Cum se folosește Java în practică, de fapt</i>	30
1.3. <i>Primul lucru pe care trebuie să îl faci</i>	30
1.4. <i>Scrie primul tău program în Java</i>	32
1.5. <i>De ce orice aplicație Java are nevoie de un emulator</i>	42
1.6. <i>Ce vei învăța mai departe</i>	46
1.7. <i>Rezumat</i>	47
2. În programele reale, obiectele nu sunt animale de companie	49
2.1. <i>Java este un limbaj orientat pe obiecte</i>	50
2.1.1. <i>Definirea atributelor obiectelor</i>	53
2.1.2. <i>Crearea instanțelor de obiect</i>	55
2.1.3. <i>Definirea comportamentelor obiectelor</i>	59
2.2. <i>Într-o aplicație reală nu crezi pisici</i>	62
2.2.1. <i>Ce înseamnă model</i>	63
2.2.2. <i>Ce este un serviciu</i>	64
2.2.3. <i>Ce este un manager</i>	65
2.2.4. <i>Ce este un proxy</i>	65
2.2.5. <i>Ce este un repository</i>	66
2.2.6. <i>Ce este un controller</i>	67
2.3. <i>Rezumat</i>	70

Partea II - Să începem cu bazele	75
3. Să începem o aplicație pe bune. Cum, deja?	77
3.1. <i>Abordarea cerințelor utilizatorilor</i>	<i>78</i>
3.2. <i>Abordarea implementării aplicației</i>	<i>80</i>
3.2.1. <i>Cum să te repeți condiționat cu „while”</i>	<i>82</i>
3.2.2. <i>Asta sau cealaltă cu „if-else”</i>	<i>88</i>
3.2.3. <i>Mai multe cazuri de decizie cu „switch”</i>	<i>93</i>
3.2.4. <i>Memorează date în structuri de tip array</i>	<i>100</i>
3.2.5. <i>Parcurge intervale definite folosind for</i>	<i>108</i>
3.3. <i>Rezumat</i>	<i>115</i>
4. Creează și tratează excepții	117
4.1. <i>Ce-i cu excepțiile</i>	<i>118</i>
4.2. <i>Cum creez și cum tratez o excepție</i>	<i>126</i>
4.2.1. <i>Folosirea excepțiilor ce nu sunt verificate la compilare</i>	<i>130</i>
4.2.2. <i>Tratarea excepțiilor</i>	<i>132</i>
4.3. <i>Erori vs. excepții</i>	<i>137</i>
4.4. <i>Erori și excepții cunoscute</i>	<i>139</i>
4.5. <i>Rezumat</i>	<i>141</i>
5. Implementarea contractelor	143
5.1. <i>Cum și de ce să folosim contracte</i>	<i>144</i>
5.2. <i>Definirea contractelor prin interfețe</i>	<i>148</i>
5.3. <i>Ascunderea completă a implementărilor contractului</i>	<i>157</i>
5.4. <i>Folosirea implementărilor implicite</i>	<i>162</i>
5.5. <i>Rezumat</i>	<i>167</i>
6. Cum și de ce să folosești colecții de date	169
6.1. <i>La ce sunt bune listele</i>	<i>169</i>
6.2. <i>Elimină duplicatele folosind seturi</i>	<i>178</i>
6.2.1. <i>Identitatea unei instanțe de obiect</i>	<i>181</i>
6.3. <i>Stochează asocieri cheie-valoare în colecții de tipul map</i>	<i>187</i>
6.4. <i>Rezumat</i>	<i>191</i>

7. Fluxuri de valori și programarea funcțională	193
7.1. <i>Ce este programarea funcțională</i>	194
7.2. <i>Tipuri clasice de interfețe funcționale</i>	196
7.3. <i>Anatomia unui flux de valori</i>	199
7.4. <i>Cele mai des folosite operații pentru fluxuri de valori</i>	200
7.5. <i>Folosirea tipului Optional</i>	209
7.6. <i>Rezumat</i>	212

Partea III - Orice aplicație reală persistă și lucrează cu date stocate 215

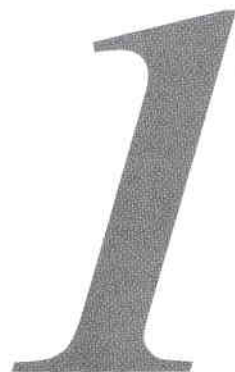
8. Stocarea persistentă a datelor în fișiere	217
8.1. <i>Cum să creezi fișiere și să obții proprietățile lor</i>	218
8.2. <i>Stocarea datelor în fișiere</i>	223
8.3. <i>Citirea datelor din fișiere</i>	229
8.4. <i>Rezumat</i>	235
9. E mai simplu să pui informații în baze de date	237
9.1. <i>Ce este de fapt o bază de date</i>	238
9.1.1. <i>Cum arată o bază de date relațională</i>	240
9.1.2. <i>Crearea unui tabel în MySQL</i>	241
9.1.3. <i>Adăugarea și regăsirea unei înregistrări</i>	247
9.2. <i>Conectarea aplicației la un server de baze de date</i>	249
9.3. <i>Implementează căutarea de date în baze de date</i>	255
9.4. <i>Implementează stocarea unei înregistrări</i>	257
9.5. <i>Folosirea obiectelor pentru administrarea conexiunilor</i>	259
9.6. <i>Rezumat</i>	262

Partea IV - Nu orice aplicație funcțională este bine scrisă 265

10. Ce este o aplicație ușor de menținut	267
10.1. <i>Păstrează simplitatea codului</i>	268
10.1.1. <i>Principiul KISS</i>	269
10.1.2. <i>Principiul DRY</i>	271
10.1.3. <i>Principiul YAGNI</i>	272

10.2. Principiile SOLID	273
10.2.1. Păstrează redus numărul de responsabilități	274
10.2.2. Fă aplicația extensibilă cu minim de modificări	276
10.2.3. Nu te baza pe mai mult decât ți se oferă	276
10.2.4. Definește contracte granulare	278
10.2.5. Bazează-te pe contracte	281
10.3. Când să fii rebel	282
10.4. Elimină împuțiciunile	283
10.4.1. Folosește denumiri intuitive	284
10.4.2. Nu defini blocuri de instrucțiuni prea lungi	284
10.4.3. Nu defini prea multe blocuri de instrucțiuni imbricate	285
10.4.4. Păstrează consistența detaliilor	286
10.4.5. Evită metode cu prea mulți parametri	287
10.4.6. O metodă trebuie, fie să producă o valoare, fie să modifice o valoare	288
10.5. Un proces corect de lucru duce la un cod mentenabil	288
10.5.1. Auditează codul (code review)	289
10.5.2. Implementează analizarea statică a codului sursă	289
10.5.3. Scrie teste automate pentru implementările tale	290
10.5.4. Comunică cu echipa ce descoperi și ce înveți nou	290
10.6. Rezumat	290
11. Cum să folosești „design patterns”	291
11.1. Cum să nu reinventezi roata	292
11.2. Învață câteva „design patterns”	292
11.2.1. Singleton	292
11.2.2. Builder	296
11.2.3. Factory method	298
11.2.4. Adapter	302
11.2.5. Decorator	306
11.2.6. Observer	311
11.3. Rezumat	317
12. Testează-ți aplicația	319
12.1. De ce să scrii teste	320
12.2. Ce teste să scrii	322
12.3. Testează unitar aplicația	322

12.3.1. Cum să testezi cazul fericit	326
12.3.2. Testează toate cazurile	332
12.3.3. Descrie cazurile testate	333
12.3.4. Verifică ce acoperire au testele tale	334
12.4. Scrie și teste de integrare	336
12.5. Rezumat	338
Appendix A Cuvinte cheie în Java	341
Appendix B Algoritmul „bubble sort”	351
Appendix C Algoritmul de sortare prin interschimbare	359
Appendix D Folosirea tipurilor generice	363
Index	375



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 îți 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. Folosește exemplele rezolvate doar atunci când ești blocat (nu înțelegi de ce nu funcționează ce ai făcut și ești 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 și de ce să înveți Java

În această secțiune aș vrea să mă asigur că știi ce este Java și de ce merită să înveți să lucrezi cu acest limbaj. Poate ești deja la curent cu faptul că Java este foarte popular. Să discutăm cele mai relevante aspecte cu privire la acest limbaj și motivele solide pentru care chiar ai vrea să începi această călătorie de studiu.

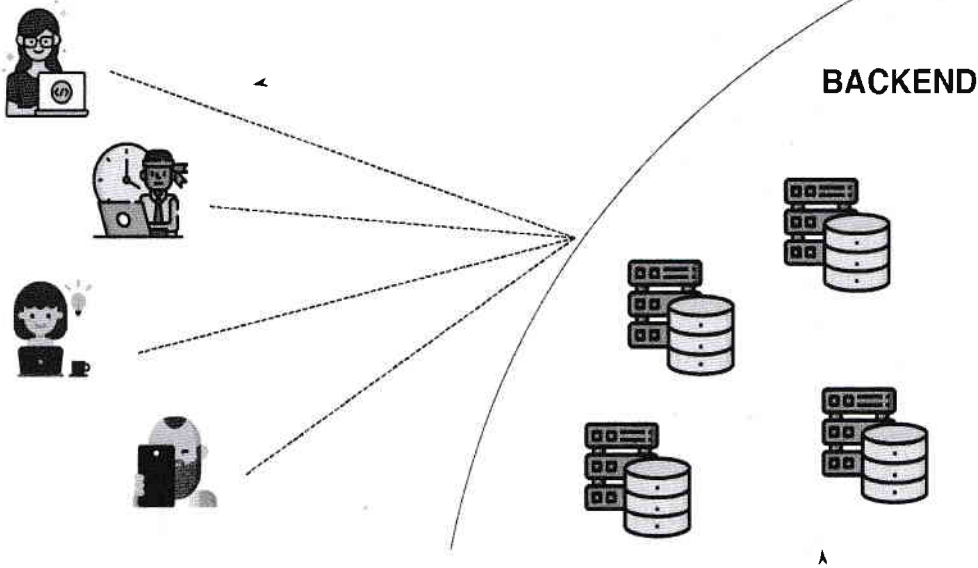
În anul 2021, conform unui sondaj realizat de Statista (<https://link.telacad.ro/java/cap1-1>), Java este pe locul 5 în topul limbajelor de programare, dar în același timp locul 1 ca limbaj de programare preferat pentru dezvoltarea aplicațiilor de tip backend.

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

În același timp, spunem că orice aplicație care interacționează direct cu utilizatorul este o componentă de frontend. În cele mai multe cazuri astăzi, componentele de frontend sunt aplicații web ce se execută prin intermediul unui browser. Aplicația ce se execută în browser când tu deschizi o pagină web este o aplicație de frontend (cel mai probabil scrisă folosind limbajul JavaScript și 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 și un număr larg de evenimente. Imaginează-ți sistemul compus din toate aplicațiile folosite de o bancă (atât cele necesare angajaților cât și cele folosite de utilizatorii băncii, precum aplicațiile de plăți 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 folosește Java în practică, de fapt?

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

Orice limbaj de programare are avantaje și dezavantaje. Avantajele și dezavantajele unui limbaj îl fac să fie destinat mai degrabă unui anumit tip de aplicații decât altul. De exemplu, așa cum vom discuta mai în detaliu în secțiunea 1.5, execuția tuturor aplicațiilor Java se bazează pe un emulator (numit și mașină virtuală, sau pe scurt JVM [Java Virtual Machine]). Acest aspect face ca aplicațiile Java să fie mai ușor de dezvoltat, însă mai puțin performante (nu fac la fel de eficient managementul procesării și uzul memoriei precum aplicațiile scrise în alte limbaje). Desigur, îți voi explica în secțiunea 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 excepție). Un limbaj precum C, sau C++ are alte caracteristici (avantaje și dezavantaje). Aplicațiile scrise cu C și C++ au execuția mult mai performantă decât cele scrise cu Java, însă, sunt mult mai dificil de scris, iar acest lucru face implementarea acestor aplicații mai costisitoare.

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

1.3. Primul lucru pe care trebuie să îl faci

În această secțiune 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 pași:

1. Să instalezi un mediu de implementare de aplicații, cunoscut și 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 și pe al doilea. Noi vom folosi un mediu de dezvoltare numit IntelliJ IDEA. Versiunea Community, pe care o poți descărca de pe site-ul oficial (<https://link.telacad.ro/java/cap1-2>) și o poți folosi gratuit este suficient pentru ce ai nevoie ca să implementezi exemplele din această carte. De fapt, IntelliJ IDEA Community este des folosit și în dezvoltarea aplicațiilor reale. Deși multe companii preferă achiziționarea unei licențe Ultimate, care adaugă capabilități 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).

Limba de Program	Obiectiv
Java	<ul style="list-style-type: none"> • Componente tip backend • Aplicații de testare automată • Mai rar, componente tip desktop (care rulează direct pe calculatorul utilizatorului)
Python, R	<ul style="list-style-type: none"> • Machine learning • Algoritmi matematici • Mai rar, implementarea componentelor de tip backend
C, C++	<ul style="list-style-type: none"> • Aplicații care trebuie să aibă o execuție rapidă (de exemplu: jocuri video) • Mai rar, aplicații de tip backend sau desktop
C#	<ul style="list-style-type: none"> • Componente tip backend • Mai rar componente tip desktop sau jocuri video
JavaScript	<ul style="list-style-type: none"> • Aplicații de tip frontend (care rulează în 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.

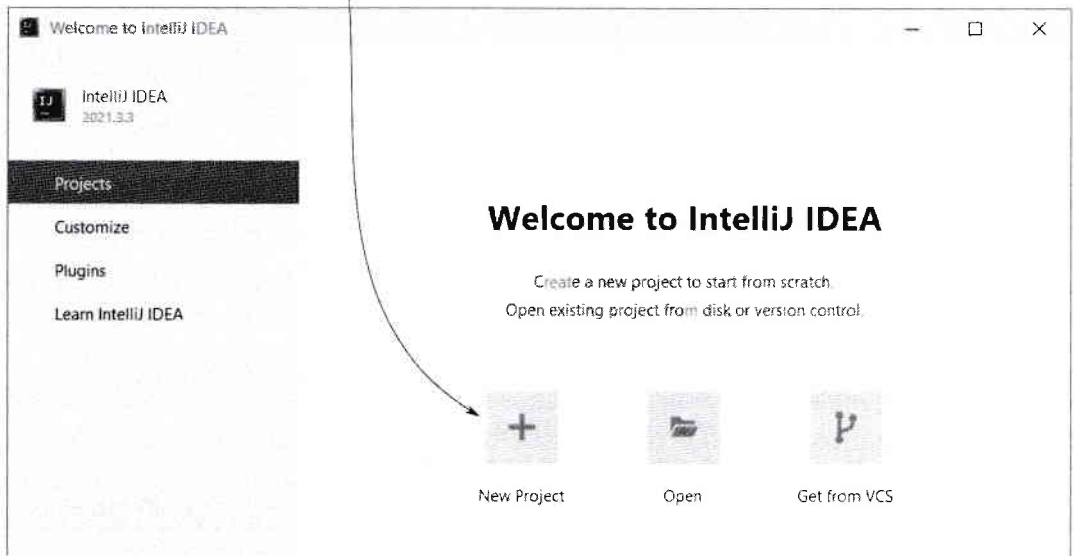


Figura 1.2 După ce ai instalat IntelliJ IDEA Community, poți crea un proiect nou folosind butonul New Project.

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

1.4. Scrie primul tău program în Java

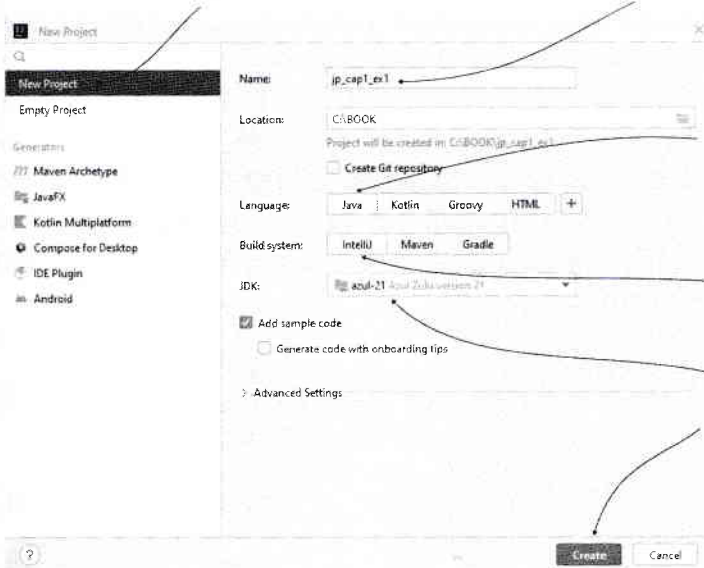
Dacă acum ai tot ce este nevoie instalat pe calculatorul tău, putem începe scrierea primului program. În această secțiune vei învăța să creezi un proiect, să scrii și să execuți primul program. Aplicația pe care o vom face va fi una simplă – un program care afișează un mesaj sub forma unui text, de exemplu „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.

1. Selectezi New Project pentru a crea un proiect nou.

2. Dai un nume proiectului.



3. Te asiguri că ai selectat limbajul Java.

4. Selectezi tipul proiectului. Momentan vom folosi proiecte simple IntelliJ.

5. Selectezi versiunea cu care vrei să lucrezi.

6. Selectezi butonul Create.

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**

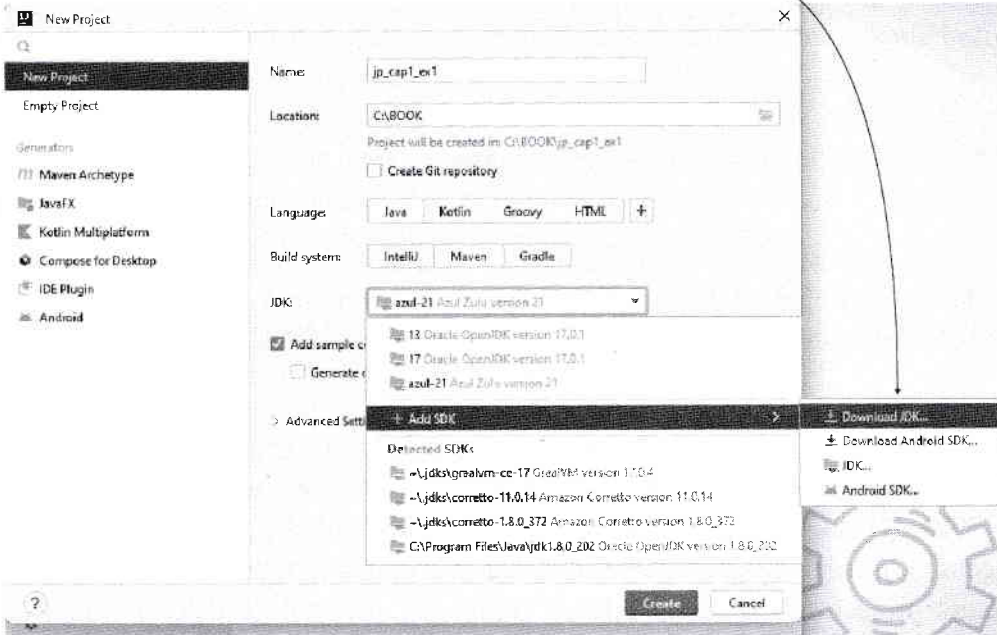


Figura 1.4 Poți descărca foarte ușor o versiune de JDK folosind direct butonul Download JDK din IDE. Ulterior descărcării, vei putea folosi această versiune a limbajului pentru a crea proiecte noi.

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>.

1. Selectezi versiunea de Java pentru care vrei să descarci kit-ul pentru dezvoltatori.

2. Opțional, selectezi o distribuție și o cale personalizată unde kit-ul de dezvoltare va fi descărcat.

3. Apeși pe **Download**.

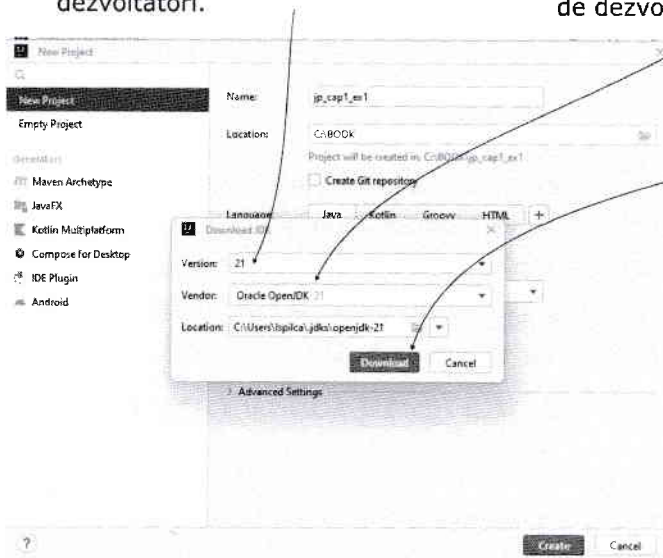


Figura 1.5 Alege versiunea de Java pe care vrei să o descarci pentru a dezvolta programe. Opțional, poți selecta o anumită distribuție și o cale personalizată unde aceasta să fie descărcată în sistemul tău.

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.

În partea din stânga a ferestrei vei găsi structura proiectului creat de tine.

Folderul „src” este cel în care vom crea fișierele ce definesc programul nostru.

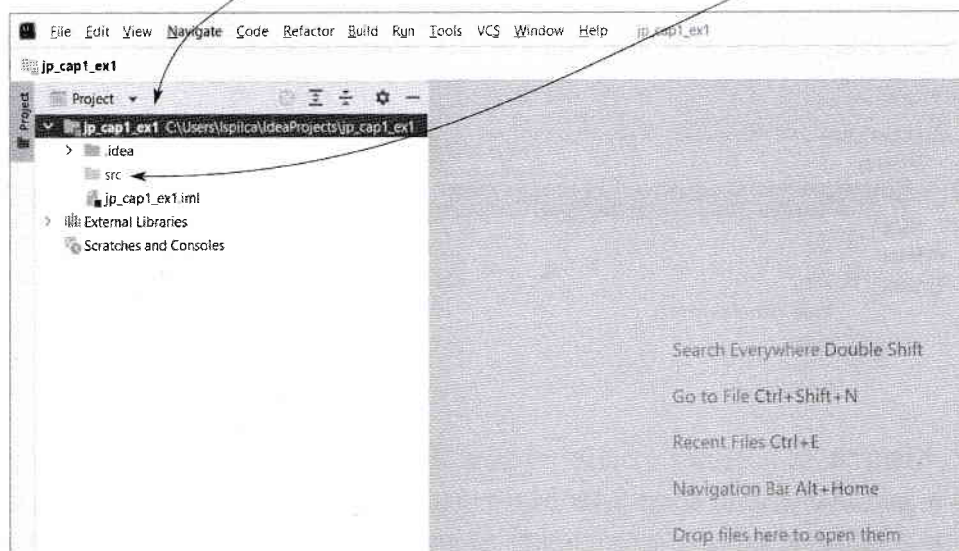


Figura 1.6 Structura proiectului. Folderul „src” este cel în care vom crea fișierele ce compun programul pe care îl scriem.

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.

Click dreapta pe folderul „src”,
 apoi alege **New > Java Class**.

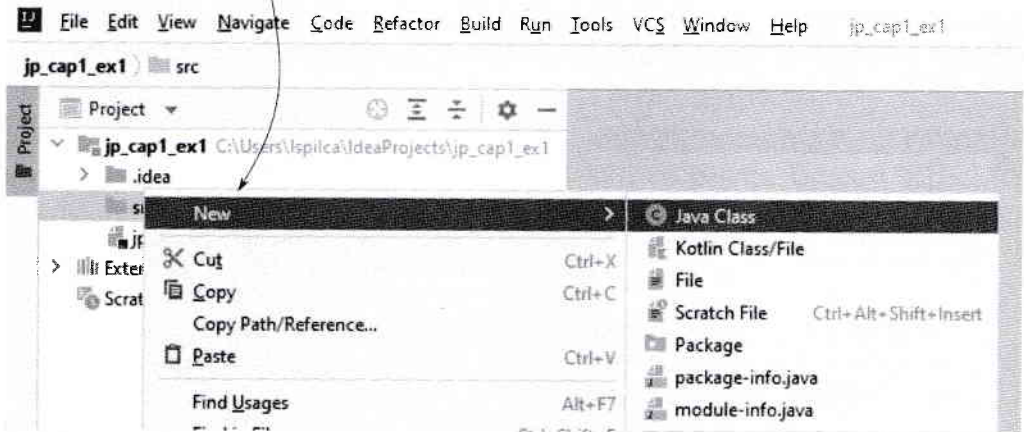


Figura 1.7 Pentru a scrie primele linii de cod, vom crea un fișier cu extensia .java. Vom numi acest fișier **Main** pentru că el este fișierul principal care definește programul nostru. Pentru a crea acest fișier, dă click dreapta pe folderul „src” din structura proiectului, apoi selectează opțiunea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

În fereastra ce apare, scrie numele fișierului pe care îl crezi. 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.

Scrie un nume pentru fișier și dă dublu click pe **Class**.

Eu îți sugerez să folosești numele **Main** pentru acest exemplu.

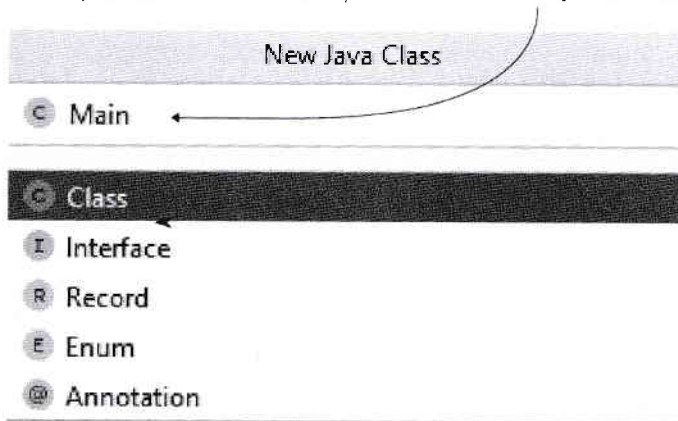


Figura 1.8 Alege numele fișierului de cod sursă și selectează opțiunea **Class** pentru a crea acest fișier.