Programare Orientată pe Obiecte: Intro

Alexandru Olteanu

Universitatea Politehnica Bucuresti
Facultatea de Automatică si Calculatoare, Departamentul Calculatoare
alexandru.olteanu@upb.ro

OOP, 2020

Ordinea de zi

- Administrativ
- De ce OOP?
- De ce Java la OOP?

Administrativ

Admin: Activitati online

Seria CD	**Comun CA+CD**
Curs, Examen	Labs, Teme
Moodle CD	Moodle CA+CD
Teams CD	Teams CA+CD
	Wiki

Admin: Reguli de desfasurare

Regulament pe Wiki

Admin: Curs vs Curs Practic

		Curs	Curs Practic	Lab
Partea I Notiuni de baza OOP	1	Intro, de ce OOP?, de ce Java si JVM?		Java Basics
	2	Ce se poate gasi intr-o clasa? specificatori de acces (1), constructori, referinte si memory model	Modele de proiecte in IntelliJ (eventual si Android Studio)	Constructori și referințe
	3	Ce relatii sunt intre clase? agregare vs compunere vs mostenire, casting, overriding, overloading, super		Agregare și moștenire
	4	Incapsulare, specificatori de acces (2), imutabilitate si final, static vs instante, Singleton	Modele de incapsulare (exemple simple, eventual Activity din Android)	Static si Final
	5	Clase abstracte și interfețe, Mostenire multipla, Factory		Clase abstracte și interfețe
	6	Clase interne	Modele de ierarhii de clase, Listeners, Inversion of Control	Clase interne
Partea a II-a Notiuni avansate OOP	7	Polimorfism, double dispatch, Visitor		Overriding, Overloading, Visitor
	8	Genericitate	Collection Framework	Colecții
	9	OO Design Principles, SOLID		Genericitate
Partea a III-a OOP Production level	10	Control Flow (Frameworks, Inversion of Control, Strategy, Observer, Chain of Responsibility)	Spring app	Excepții
	11	More on frameworks (Entities, Repository, POJO, CRUD)		Design Patterns
	12	Breaking Patterns: Antipatterns, Extending without Inheritance (Decorator, Prototype), Reflection	Testarea codului, Junit si Mock, Test Driven Development	Design Patterns

De ce OOP

Exercitiu: OOP vs Programare Procedurala

Comparati varianta C si varianta C++ a aceluiasi program:
Exercitiu pe moodle

Paradigme de Programare

- imperativă (programatorul descrie pas cu pas rezolvarea problemei)
  - procedurală
  - orientată-obiect
  - etc.
- declarativă (programatorul descrie proprietati ale rezultatului dorit, nu cum se calculeaza)
  - functională
  - logică
  - etc.

Nota: limbajele de programare pot avea trasaturi din mai multe paradigme de programare (e.g. in Java 8+ avem clase si expresii lambda)
Programming Paradigms, Ray Toal

Programare Orientată-Obiect vs Programare Procedurală

Programare Procedurală	Programare Orientată-Obiect
Procedură: O listă de instrucțiuni care îi spun computerului ce să facă pas cu pas	Obiect: componentă a programului care știe cum să desfășoare anumite acțiuni și cum să interacționeze cu alte elemente din program

Programare Orientată-Obiect vs Programare Procedurală

Programare Procedurală	Programare Orientată-Obiect
Procedură: O listă de instrucțiuni care îi spun computerului ce să facă pas cu pas	Obiect: componentă a programului care știe cum să desfășoare anumite acțiuni și cum să interacționeze cu alte elemente din program
• tipuri de date primitive • tipuri de date compuse (a.k.a. composite data type, a.k.a. record, e.g. struct in C)	• tipuri de date primitive • tipuri de date compuse (struct in C) • clase (tipuri ce compun date si actiuni)

Programare Orientată-Obiect vs Programare Procedurală

C	printf("Hello World!\n")	printf este o functie

Programare Orientată-Obiect vs Programare Procedurală

C	printf("Hello World!\n")	printf este o functie
Java	System.out.println("Hello World!")	System.out este un obiect, println este metoda sa

Programare Orientată-Obiect vs Programare Procedurală

C	printf("Hello World!\n")	printf este o functie
Java	System.out.println("Hello World!")	System.out este un obiect, println este metoda sa
C++	cout « "Hello World!" « endl	

Programare Orientată-Obiect vs Programare Procedurală

C	printf("Hello World!\n")	printf este o functie
Java	System.out.println("Hello World!")	System.out este un obiect, println este metoda sa
C++	cout « "Hello World!" « endl	cout este un obiect, operator« este metoda sa

Programare Orientată-Obiect vs Programare Procedurală

Programare Procedurală	**Programare Orientată-Obiect**
Procedură: O listă de instrucțiuni care îi spun computerului ce să facă pas cu pas	Obiect: componentă a programului care știe cum să desfășoare anumite acțiuni și cum să interacționeze cu alte elemente din program
simplu, reflectă un mod direct de a rezolva problema, ok pentru programe mici	avansat, modelează situații din realitate, oferind modularitate și ordine în programele mari

Cu ce ajută un design orientat obiect?

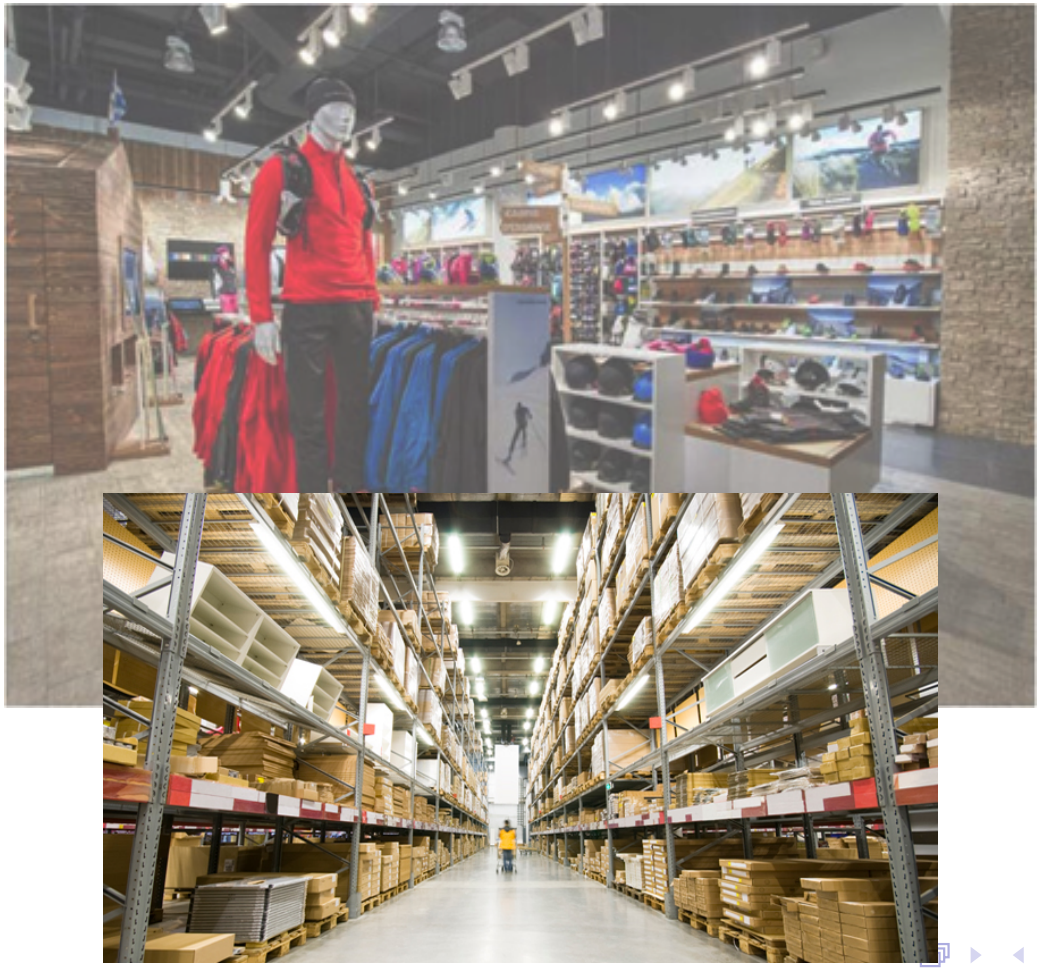
- scrierea de cod complex devine clară și cu minim de erori (readibility)
- împărțirea codului între membrii echipei si urmarirea progresului (transparency)
- cod ușor de extins și reparat (extensibility)
- devine clar ce cod este testat automat si ce cod nu este testat (testability)
- refolosim soluții la probleme comune și uneori chiar cod (reusability)
- usurinta in crearea - uneori automata - a documentatiei (documentation)
- evităm greșelile făcute de alții

Reutilizarea codului

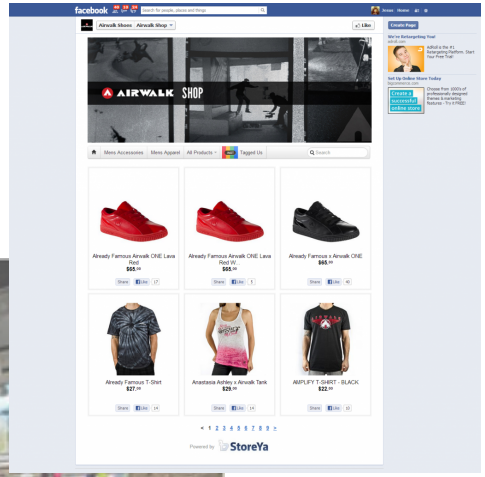
Reutilizarea conceptelor

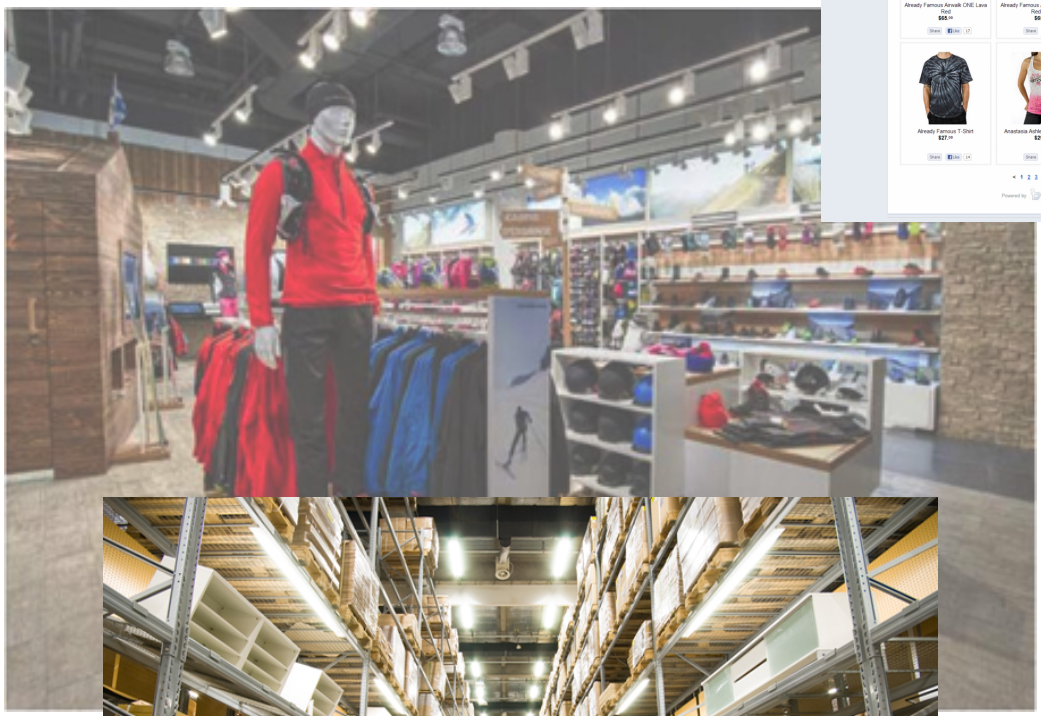
Complexitatea aplicațiilor

Complexitatea aplicațiilor

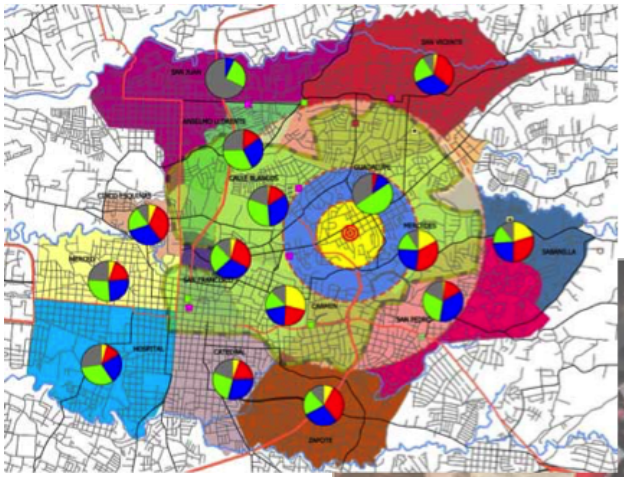


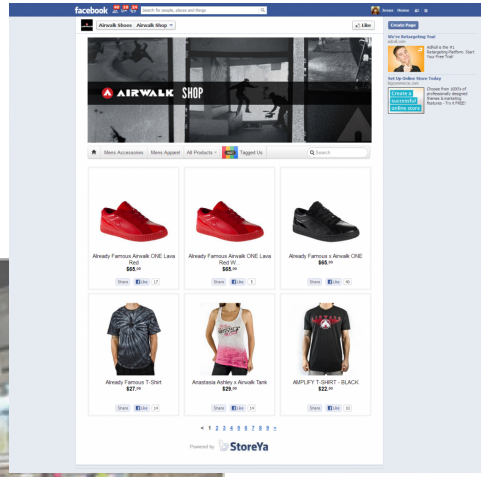
Complexitatea aplicațiilor





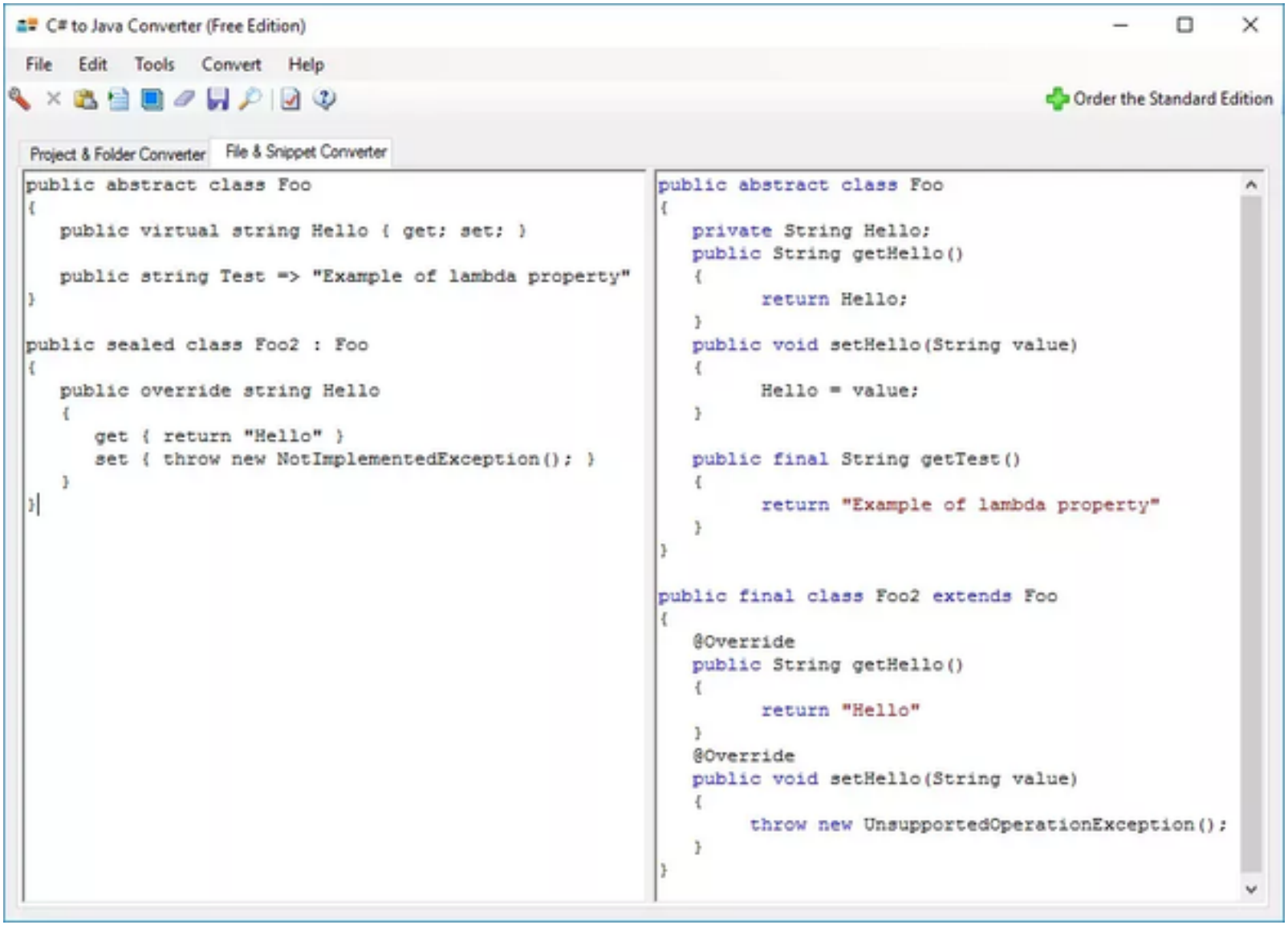
Complexitatea aplicațiilor





De ce JVM si Java?

Usurinta de a trece de la un limbaj la altul



sursa

Java Virtual Machine (JVM)

Java Virtual Machine:

- JVM: mașina virtuală ce permite transformarea codului intermediar în instrucțiuni executabile pe procesorul curent (a.k.a. un program ce execută cod, proiectat pentru portabilitate, trade-off de performanta)

Java Virtual Machine (JVM)

Java Virtual Machine:

- JVM: mașina virtuală ce permite transformarea codului intermediar în instrucțiuni executabile pe procesorul curent (a.k.a. un program ce execută cod, proiectat pentru portabilitate, trade-off de performanta)
- Mai multe limbaje de programare folosite pentru a produce cod ce ruleaza in JVM: Java, Scala, Kotlin, Clojure, Groovy etc. ▸ 20+ limbaje

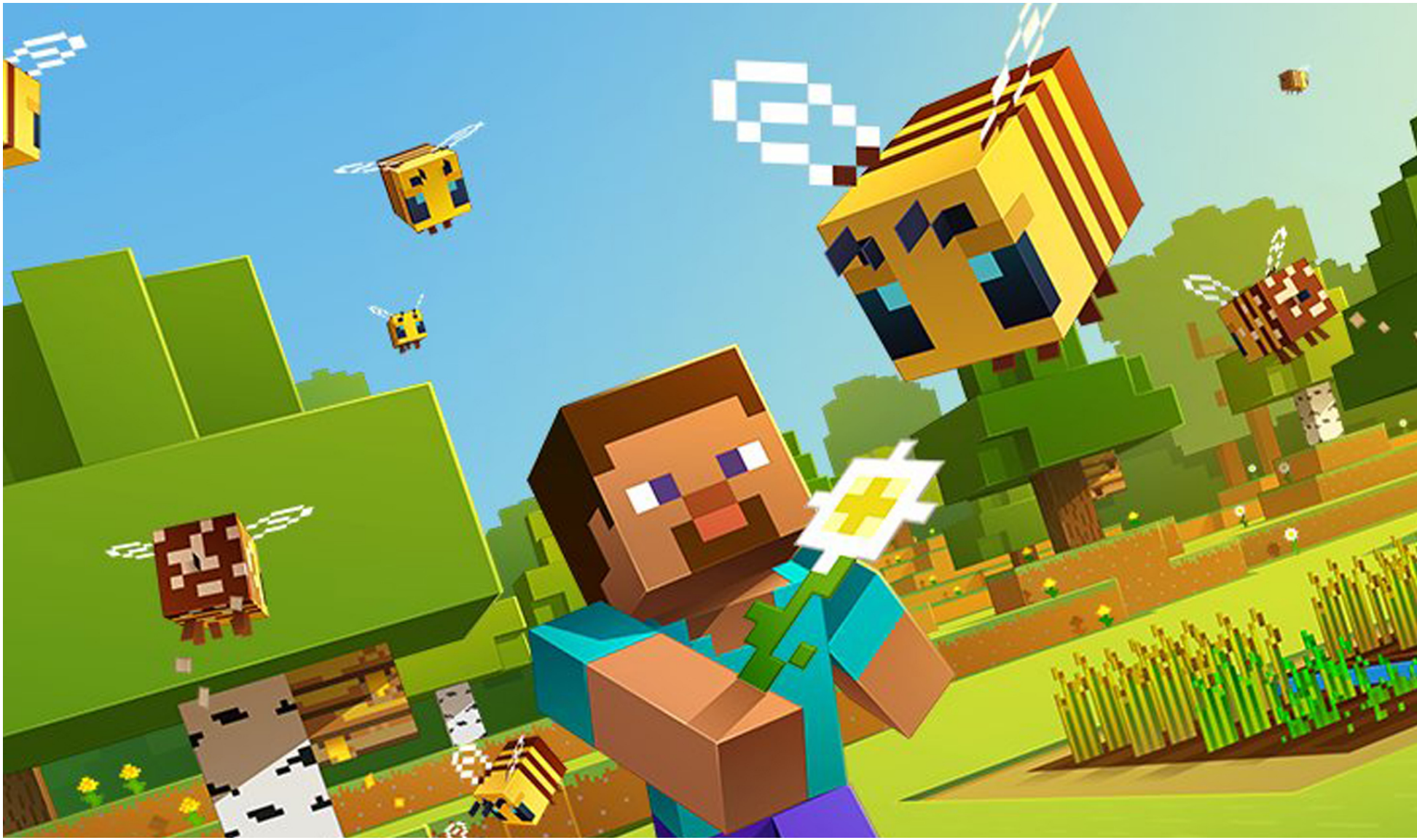
Cu ce ajută Java la OOP-CD?

- unul dintre limbajele de programare majore (exemplu de clasificare)
- gestiunea usoara a memoriei (fata de C++)
- focus pe OOP (fata de Python si JavaScript)
- putem sa luam exemple de Android si Spring

Java uses: Lucene, Elasticsearch

sursa

Java uses: first version of Minecraft



sursa

Java uses: Maestro Science Activity Planner

sursa

Java

Java platform editions:

- Java SE: basics, networking, security, database access, GUI
- Java EE: developing and running large-scale network applications
- Java ME: small-footprint virtual machine for small devices
- Java FX: use hardware-accelerated graphics and media engines

Java

Java platform editions:

- Java SE: basics, networking, security, database access, GUI
- Java EE: developing and running large-scale network applications
- Java ME: small-footprint virtual machine for small devices
- Java FX: use hardware-accelerated graphics and media engines

Java environments:

- JRE: Java Runtime Environment
- JDK: Java Development Kit (include JRE)

Reading Assignments

- Lab 1: Java Basics
- If everyone hates it, why is OOP still so widely spread?

Anexa 1: Recommended Reading Material

- SCJP Sun Certified Programmer for Java™ 6 - Study Guide
  K. Sierra, B. Bates, McGraw Hill, 2006.
- Effective Java
  J. Bloch, Addison Wesley, 2008.
- Thinking in Java
  B. Eckel, Prentice Hall, 2006.
- Design Patterns
  E. Gamma, R. Helm, R. Johnson, J. Vlissides, Addison Wesley, 1994.
- Code Complete
  S. McConnell, Microsoft Press, 2004.

Anexa 2: Situarea cursului

Soluții Business							**AII**
Grafică					**EGC**		**SPG**
Metaprogramare					**LFA**		**CPL**
Embedded						**PM**	**SI**
Prog.Par&Distr					**APD**	**ASC**	**APP**
Comunicație				**PC**			**SPRC**
Algoritmică			**AA**	**PA**			**IA**
Bazele Prog.	**PC**	**SD**	**POO**	**PP**			
	An 1 Sem1	An 1 Sem2	An 2 Sem1	An 2 Sem2	An 3 Sem1	An 3 Sem2	An 4 Sem1