NACIONĀLAIS ATTĪSTĪBAS PLĀNS 2020
EIROPAS SAVIENĪBA
Eiropas Sociālais fonds
IEGULDĪJUMS TAVĀ NĀKOTNĒ

Java programmēšanas pamati

15. NODARBĪBA

KONTROLES INVERSIJA

«INVERSION OF CONTROL» PRINCIPS

Elastīgas sistēmas ir tās, kurās izejas koda atkarība norāda uz abstrakciju, nevis konkrētu implementāciju

jeb praksē

Objektu kontrole (izveide) vai pat programmas daļas pārvaldība tiek nodota citām klasēm «no ārpuses»

IOC PRAKSĒ

- Atsaucēs neizmanto mainīgu konkrētu klasi – izmanto abstrakciju jeb atsaucies uz interfeisu
- Neatvasini no mainīgas konkrētas klases
- Nepārraksti (override) konkrētas klases metodes
- Nekad nepiemini neko konkrētu vai mainīgu – IOC princips citiem vārdiem

IOC PRIEKŠROCĪBAS

- Atdala uzdevumu izpildi no tā implementācijas
- Padara ērtu pārslēgšanos starp dažādām implementācijām
- Lielāka programmas modularitāte
- Vieglāk testējams kods, izolējot komponenti vai izmatojot «mokus» aizstāt tās atkarības

KĀ IMPLEMENTĒT

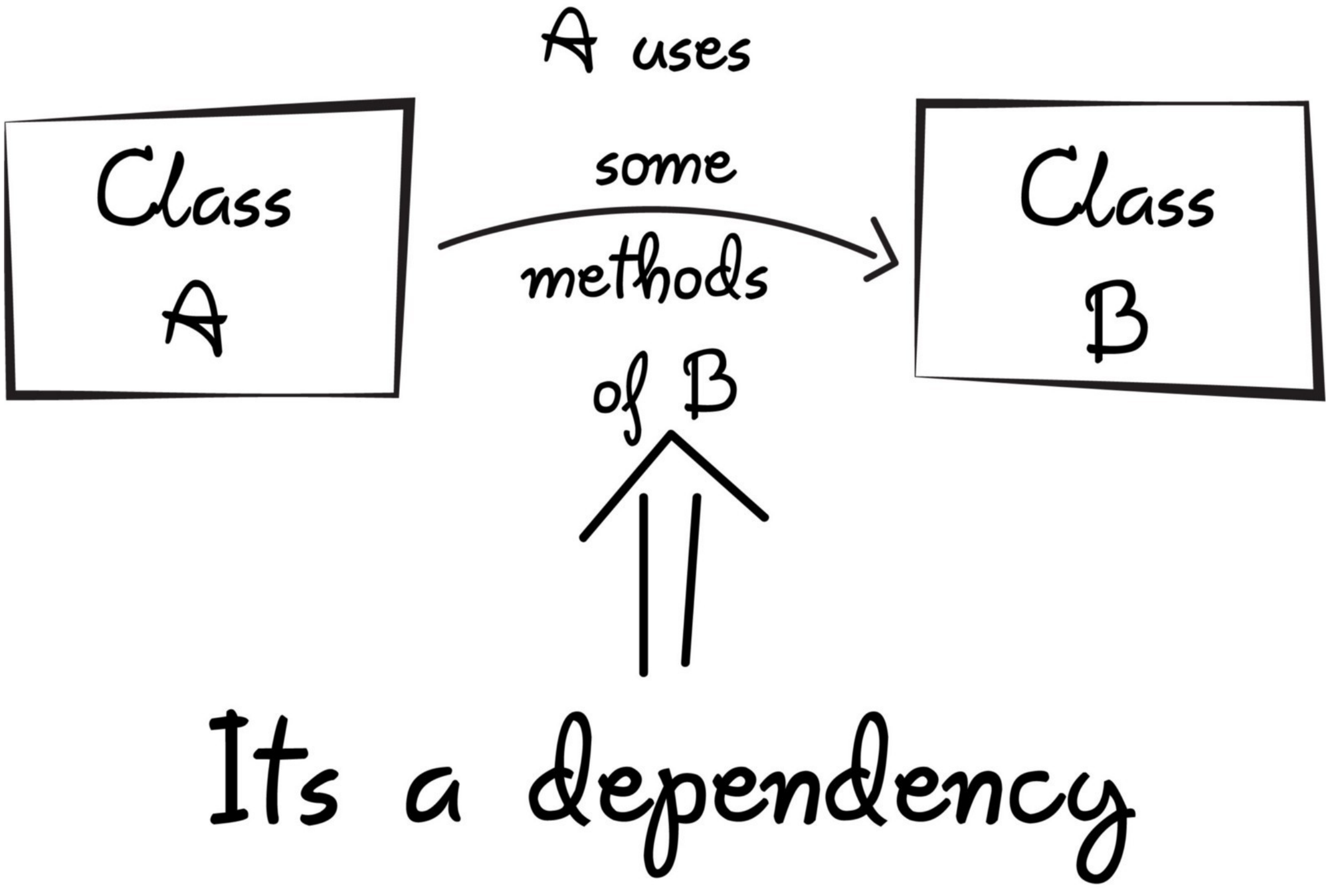
- Stratēģijas dizaina tehnika (Strategy)
- Servisa atradēja dizaina tehnika (Service locator)
- Rūpnīcas dizaina tehnika (Factory)
- Atkarību injekcijas dizaina tehnika (Dependency Injection (DI))

ATKARIBAS
INJEKCIJA

ATKARĪBU INJEKCIJA (AI)

Atkarību injekcija ir programmēšanas tehnika, kas padara klasi neatkarīgu no tās atkarībām (dependencies). Tas tiek panākts atdalot objekta izmantošanu no tā izveidošanas

KLASEI A IR ATKARĪBA NO KLASES B



KĀPĒC IZMANTOT (AI)

- AI tehnika ļauj mainīt klases atkarības programmas darbības laikā
- Klase kļūst neatkarīga no tās izmantotajiem objektiem

AI TIPI

- **Konstruktora injekcija** – atkarības tiek nodotas, izmantojot klases konstruktoru
- **Setera injekcija** – atkarības tiek nodotas, izmantojot seteru metodi
- **Interfeisa injekcija** – atkarības klasei ir jābūt injekcijas metodei, kas iespraudīs šo atkarību jebkuram klientam. Piemēram: Klientam ir jāimplementē interfeiss, kurš atver setera metodi, kas iesprauž šo atkarību

(1) AI - PIEMĒRS

```
public class Car {
    private Wheels wheel = new MRFWheels();
    private Battery battery = new ExcideBattery();
    //...
}
```

(2) AI - PIEMĒRS

```
public class Car {
    private Wheels wheel;
    private Battery battery;

  /*
    Kādā citā vietā mūsu kodā, mēs inicializējam klasei vajadzīgos objektus.
    Šeit ir divu veidu AI implementācijas:
    1. Konstruktora tipa
    2. Setera tipa
   */

    // Konstruktora tipa
    Car(Wheel wheel, Battery battery) {
        this.wheel = wheel;
        this.battery = battery;
    }

    // Setera tipa
    void setWheel(Wheel wheel) {
        this.wheel = wheel;
    }

}
```

ATKARĪBAS INJEKCIJA

AI ATBILDĪBA

1. Izveidot objektus
2. Zināt, kurām klasēm šie objekti ir nepieciešami
3. Nodrošināt šīs klases ar objektiem

ATSAUCES

- https://martinfowler.com/articles/injection.html
- https://martinfowler.com/bliki/InversionOfControl.html
- https://blog.cleancoder.com/uncle-bob/2012/08/13/the-clean-architecture.html
- https://www.goodreads.com/book/show/18043011-clean-architecture
- https://medium.freecodecamp.org/a-quick-intro-to-dependency-injection-what-it-is-and-when-to-use-it-7578c84fa88f

LATVIJAS UNIVERSITATE
ANNO 1919

VADĪBAS UN UZŅĒMĒJDARBĪBAS