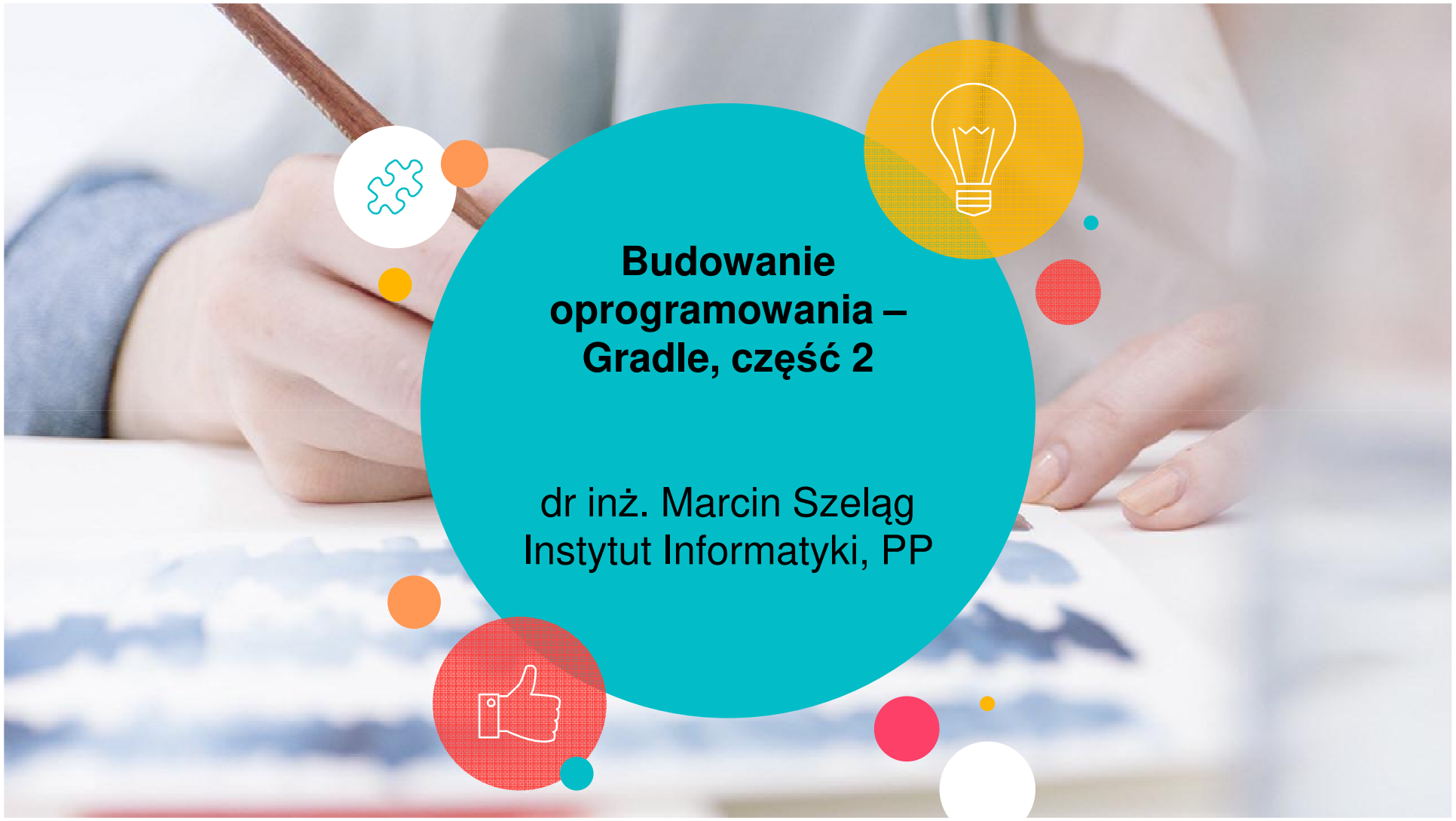


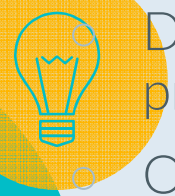
Budowanie oprogramowania – Gradle, część 2

dr inż. Marcin Szeląg
Instytut Informatyki, PP

Gradle – API

- Każdy skrypt budujący (`build.gradle`) odpowiada niejawnej instancji obiektu typu `Project`
- **Gradle API:** https://docs.gradle.org/current/javadoc - najważniejszy interfejs: **`Project`**, oferujący dostęp do wszystkich funkcjonalności Gradle'a (kluczowy opis w sekcji Dynamic Project `Properties`)
- Kluczowe informacje odnośnie **powiązania identyfikatorów w skrypcie budującym z API Gradle'a (dla Groovy DSL)**: `https://docs.gradle.org/current/userguide/groovy_build_script_primer.html`

Gradle – własności (ang. properties)

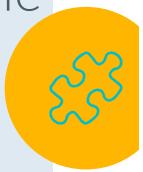


Dlaczego mowa jest o **własnościach** (properties) projektu, skoro `Project` jest interfejsem?

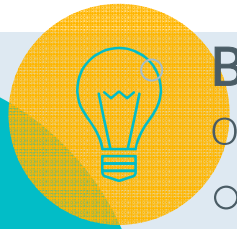
Odpowied**ź** – język Groovy oferuje **pseudo-własności**! -> ostatni przykład w opisie http://groovy-lang.org/objectorientation.html#properties

Własności są implementowane jako metody; mogą być zmieniane gdy jest setter, czytane gdy jest getter, np.:

- `println project.version` spowoduje wywołanie `project.getVersion()`
- `project.version = '0.1.0'` spowoduje wywołanie `project.setVersion('0.1.0')`



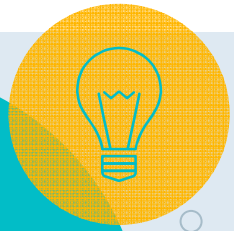
Gradle – bloki



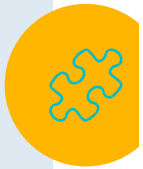
Blok = wywołanie metody ze specyficznym typem ostatniego argumentu (`groovy.lang.Closure` lub `org.gradle.api.Action`), np.:

- `<obj>.<name>(<arg>, <arg>) {...}`
-
```
project(':util') {
    apply plugin: 'java-library'
}
```
- `subprojects {...}`
- W ramach bloku może obowiązywać inna (niż do projektu) **delegacja niekwalifikowanych metod i własności** (https://docs.gradle.org/current/userguide/groovy_build_script_primer.html#delegation)

Gradle – bloki

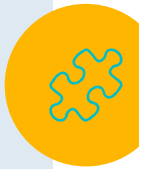


- Każdy **blok odpowiada metodzie obiektu**, w którego kontekście jest zdefiniowany
- typ **ostatniego parametru** tej metody to `groovy.lang.Closure` lub `org.gradle.api.Action` (zalecany wariant)



Gradle Build Language *

- Gradle Build Language = Gradle domain specific language = Gradle **DSL**
- wykonanie skryptu Gradle'a = **konfiguracja delegowanego obiektu Gradle'a** (skrypt `settings.gradle` -> obiekt `Settings`, skrypt `build.gradle` -> obiekt `Project`)
- **skrypt**: **instrukcje** (definicje zmiennych, metod i klas, przypisania zmiennych/własności, wywołania metod) + **bloki** (wywołania metod z `Closure` lub `Action` jako ostatnim parametrem)



* https://docs.gradle.org/current/dsl

Gradle Build Language *

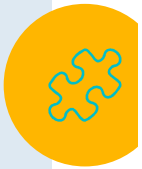
Gradle Build Language Reference:

- **Bloki** –np. allprojects, buildscript, dependencies, repositories, subprojects

  -> sekcja **`Build script blocks`**

- **Główne typy**

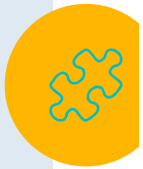
  -> sekcja **`Core types`**



* https://docs.gradle.org/current/dsl

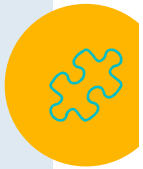
Gradle – gł. typy

- **Project**
- **Task** -> akcje, zależności, uporządkowanie
- Settings
- Gradle
- Configuration
- Script



Gradle – zarządzanie konfiguracjami *

- **Konfiguracja** (dependency configuration)– określa zakres widoczności (ang. scope) zależności (ang. dependencies)
- Identyfikowana jest przez unikalną nazwę
- Może zostać dodana do projektu przez plugin
- Np. konfiguracja `implementation` (plugin `Java`) grupuje zależności, które są potrzebne na ścieżce (classpath) podczas kompilacji kodu źródłowego Java
- Możliwość def. własnych konfiguracji
- Konfiguracje mogą dziedziczyć zależności

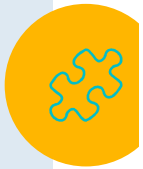


* https://docs.gradle.org/current/userguide/declaring_dependencies.html#sec:what-are-dependency-configurations

Gradle – rozszerzenia języka Groovy

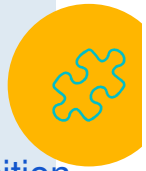
Dlaczego poniższy kod w build.grade działa?

```
task hello {
    doLast  {
        println 'Hello world!'
    }
}
hello.doLast  {
    println "Greetings from the $hello.name task."
}
```



Gradle – rozszerzenia języka Groovy

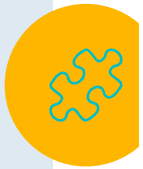
- Gradle definiuje klasy dokonujące **transformacji drzewa składniowego** (AST = abstract syntax tree) programu w języku Groovy na etapie kompilacji (AST Transformations); jest to mechanizm oferowany przez Groovy'ego
- Dzięki temu, skrypty gradle mogą zawierać dodatkowe konstrukcje językowe, które są tłumaczone na język Groovy dopiero podczas kompilacji



-https://stackoverflow.com/questions/27584463/understanding-the-groovy-syntax-in-a-gradle-task-definition
-http://groovy-lang.org/metaprogramming.html#_compile_time_metaprogramming
-https://github.com/gradle/gradle/blob/master/subprojects/core/src/main/java/org/gradle/groovy/scripts/internal/TaskDefinitionScriptTransformer.java – klasa opisująca transformację fragmentów skryptu Gradle'a odpowiedzialnych za definicję zadań, takich jak np. `task hello`

Gradle – tworzenie zadań

- **Podstawowe informacje** dot. zadań: https://docs.gradle.org/current/userguide/tutorial_using_tasks.html
- **Zaawansowane zadania** (ang. enhanced tasks): https://docs.gradle.org/current/userguide/more_about_tasks.html
- **Tworzenie własnych zadań** (przykład własnego zadania Gradle'a rozszerzającego klasę `DefaultTask`): https://guides.gradle.org/writing-gradle-tasks



Gradle – tworzenie pluginów

- Klasa **pluginu** implementuje interfejs `Plugin<Project>`
- https://guides.gradle.org/designing-gradle-plugins/