# Wybrane elementy praktyki projektowania oprogramowania
# Zestaw 8

Projektowanie obiektowe, UML

2018-01-09

Liczba punktów do zdobycia: **10/80**
Zestaw ważny do: 16-01-2018

*Uwaga! W zadaniach w których mowa jest o przedstawieniu wybranego diagramu UML, należy użyć* **jakiegoś** *narzędzia typu CASE - na wykładzie rekomendowano Enterprise Architect i Visual Paradigm for UML (ten ostatni ma darmową wersję Community Edition i działa na wielu systemach operacyjnych). Pośrednim celem zadań jest również bowiem zapoznanie się ze współczesnym warsztatem architekta oprogramowania - stąd wymaganie dedykowanego oprogramowania, a nie np. edytora graficznego w którym przy odrobinie wysiłku diagramy też można próbować rysować. Proszę więc zapomnieć o oprogramowaniu typu Paint ale też np. o Dia, StarUML, draw.io i innych narzędziach w których tworzenie diagramów jest technicznie możliwe, acz karkołomne.*

1. (**2p**) Zdokumentować dwa przypadki użycia wybranego przez siebie przykładowego problemu (gra w brydża, zakupy w sklepie internetowym, inne). Co najmniej jeden opisać w formie skróconej (*brief*) i jeden w formie pełnej (*fully dressed*).

   *Uwaga!* Przykład opisu w formie pełnej:

   `https://jira.atlassian.com/secure/attachment/48985/Use+case+POS.pdf`

2. (**2p**) Zbudować w postaci diagramu UML model pojęciowy dla wybranego przez siebie problemu. Zwrócić uwagę na identyfikację atrybutów oraz asocjacji. Model nie powinien być mniejszy niż 5 i większy niż kilkanaście pojęć. Każde pojęcie powinno być związane z co najmniej jednym innym pojęciem. Nad asocjacjami dopisać nazwy przedstawiające ich znaczenia.

   Uszczegółowić diagram modelu pojęciowego i przedstawić diagram klas odpowiadający diagramowi modelu pojęciowego.

   Oba diagramy, diagram modelu pojęciowego i diagram klas przedstawić w postaci UML.

   Pokazać **różnice** między diagramami i nauczyć się je odróżniać.

3. (**2p**) Przedstawić diagram klas UML dla poniższego kodu:

```
public interface ICommand
{
    void Execute( string CommandName );
}

public abstract class AbstractCommand : ICommand
{
    private int commandCount;
```

```
    protected string commandState;
    public string commandName;

    private void commandBuilder() { }
    public abstract void Execute( string CommandName );
}

public class ConcreteCommand : AbstractCommand
{
    protected CommandStepBuilder commandBuilder;

    public override void Execute( string CommandName ) { }
}

public class CommandStepBuilder
{
    public const int MAXSTEPS = 10;
    public static int StepCount;
}
```

4. (**2p**) Przestawić diagram czynności UML opisujących interakcję użytkownika z prostym urządzeniem typu bankomat (nie więcej niz kilkanaście akcji). Przewidzieć jakieś sytuacje wyjątkowe (brak gotówki, błędnie wprowadzona kwota, błędny PIN), skutkujące pojawieniem się zdarzeń.

5. (**2p**) Zdokumentować w postaci diagramu sekwencji UML proces rejestracji nowego konta w przykładowej usłudze internetowej. Zdefiniować co najmniej użytkownika systemu i dwóch różnych uczestników procesu (różne typy odpowiedzialności): interfejs użytkownika i repozytorium danych.

Wiktor Zychla