**INŻYNIERIA OPROGRAMOWANIA PROJEKT – DOKUMENTACJA I OPIS**

**Zadanie**

Celem zadania projektowego będzie stworzenie działającego i w pełni funkcjonalnego oprogramowania.

**Źródła:**

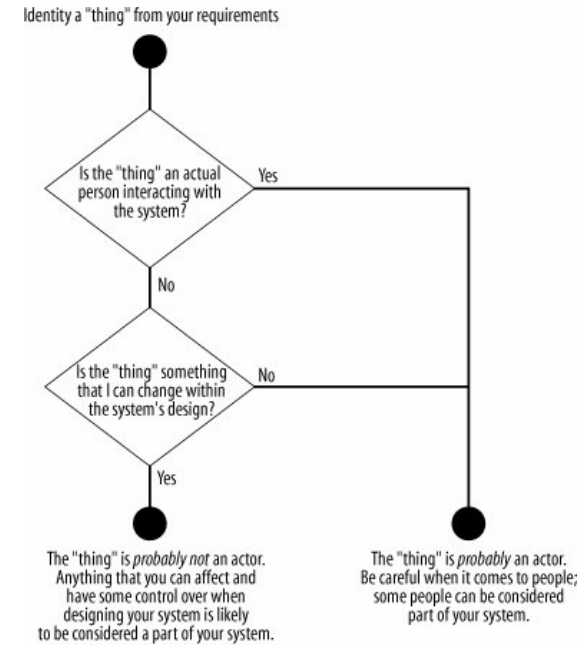
**1. Learning UML 2.0 A Pragmatic Introduction to UML** Russ Miles, Kim Hamilton

Wydawca: O'Reilly Media 1 edition (5 May 2006)

ISBN-13: 978-0596009823

Wykorzystane fragmenty z rozdziału 2:

1.1 Wybieranie aktorów, strona 44

****

1.2 Wybieranie przypadków użycia, strona 46

“Once you have captured an initial set of actors that interact with your system, you can assemble the exact model of those interactions. The next step is to find cases where the system is being used to complete a specific job for an actoruse cases, in fact. Use cases can be identified from your user's requirements.”

1.3 Linie komunikacji, strona 47

“The purpose of a communication line is to show that an actor is simply involved in a use case, not to imply an information exchange in any particular direction or that the actor starts the use case. That type of information is contained within a use case's detailed description, therefore it doesn't make sense to apply navigation to communication lines.

A use case is something that provides some measurable result to the user or an external system. Any piece of system behavior that meets this simple test is likely to be a good candidate for a use case.”

1.4 Relacja <<include>>, podrozdział 2.2.1, strony 53 – 55

1.5 Granice systemu, strona 48

“To show your system's boundary on a use case diagram, draw a box around all of the use cases but keep the actors outside of the box. It's also good practice to name your box after the system you are developing.”

**2. Pragmatic Unit Testing in Java with JUnit** 1st Edition Jeff Langr, Andy Hunt and Dave Thomas

Wydawca: Pragmatic Bookshelf 16 marca 2015

ISBN-13: 978-0974514017

Wykorzystany fragment, rozdział 8 strona 99

**“**Where to Put Test Code, "In general, you don't want to break any encapsulation for the sake of testing (or as Mom used to say, "don't expose your privates!"). Most of the time, you should be able to test a class by exercising its public methods. If there is significant functionality that is hidden behind private or protected access, that might be a warning sign that there's another class in there struggling to get out."

**3**. **JavaFX:** [https://docs.oracle.com/javafx/2/best\_practices/jfxpub-best\_practices.htm](https://l.facebook.com/l.php?u=https%3A%2F%2Fdocs.oracle.com%2Fjavafx%2F2%2Fbest_practices%2Fjfxpub-best_practices.htm&h=ATOZrIk3eAP9-3b8yLD5g_pPBrbAJONVJbOs19KMRYCM_GlF4hWJ2eM5M___iizytdstKG9o4qNh7y7tuV9H7951O8X9k1fg4kMwYYt20FoZ0riA70EBWInR1Q3RMVlyLfcw3L_S2yVlDQtO)

Dobre praktyki JavaFX

**4**. **JavaFX:** [https://docs.oracle.com/javafx/2/get\_started/jfxpub-get\_started.htm](https://l.facebook.com/l.php?u=https%3A%2F%2Fdocs.oracle.com%2Fjavafx%2F2%2Fget_started%2Fjfxpub-get_started.htm&h=ATOZrIk3eAP9-3b8yLD5g_pPBrbAJONVJbOs19KMRYCM_GlF4hWJ2eM5M___iizytdstKG9o4qNh7y7tuV9H7951O8X9k1fg4kMwYYt20FoZ0riA70EBWInR1Q3RMVlyLfcw3L_S2yVlDQtO)

Przewodnik po JavaFX.

**5**. **SceneBuilder:** <https://docs.oracle.com/javase/8/scene-builder-2/user-guide/index.html> - Przewodnik po Scene Builderze.

**6**.**Visual Paradigm** <https://www.visual-paradigm.com/support/documents/vpuserguide/94/2575/6362_drawinguseca.html>

Opis tworzenia diagramu przypadków użycia w Visual Paradigm.

**7. Java SE** <https://docs.oracle.com/javase/9/>

Dokumentacja Javy 9

**Dobrane narzędzia oraz technologie:**

1. Visual Paradigm –<https://ap.visual-paradigm.com/agh-university-of-science-and-technology2>
2. Repozytorium Git - <https://github.com/>
3. IntelliJ IDEA: The Java IDE for Professional Developers by JetBrains - <https://www.jetbrains.com/idea/>
4. Język programowania – Java
5. Scene Builder http://gluonhq.com/products/scene-builder/