

Programowanie zespołowe

dr inż. Marcin Ochab
mjochab@ur.edu.pl

www: <http://mjochab.edu.pl>

22.02.2021

Wstep

Zasady ogólne

- ▶ Tok indywidualny:
 - * wszystkie zasady pozostają takie same
 - * **łącznie z obecnościami na laboratoriach!**
- ▶ Obecności (także wykład):
 - * dopuszczalne 2 nieusprawiedliwione nieobecności
 - * **3 nieobecność = skreślenie z listy studentów**
 - * nie ma możliwości odrabiania
 - * **nieprzygotowanie się do zajęć = nieobecność!**

Zaliczenie

- ▶ Ukończony projekt oddany w nieodwołalnym terminie:
 - * ostatnie zajęcia: **9.06.2021**
 - * **niedotrzymanie daty = powtarzanie przedmiotu!**
- ▶ 5 kolokwiów:
 - * podstawy Javy - 3.III.2021
 - * GIT z linii komend - 24.III.2021
 - * SQL - 14.IV.2021
 - * standardy kodowania - 28.IV.2021
 - * zaawansowana Java - 26.V.2021
- ▶ Dwie poprawki każdego - hurtowo dopiero w sesji podstawowej i poprawkowej

Zaliczenie

- ▶ prezentacja* przez grupę jednego z referatów:
 - ▶ budowanie złożonych kwerend SQL (31.III.2021)
 - ▶ dokumentowanie kodu źródłowego – javadoc (12.V.2021)
 - ▶ standardy kodowania (21.IV.2021)

*prezentacja następuje po uprzednim zatwierdzeniu jej treści i jakości przez prowadzącego (ok 30 min)

- ▶ Spełnienie zasad SCRUM!
- ▶ Pozytywne zaliczenie **każdego** z wymagań
- ▶ Zaliczenie wykładu **tylko** po zaliczeniu projektu

Kryteria oceny

Ocena końcowa jest wyznaczana na podstawie:

- ▶ Przedmiot lab. - średnia arytmetyczna ocen z zaliczonych pisemnych kolokwii - student musi zaliczyć na ocenę pozytywną wszystkie kolokwia
- ▶ Przedmiot proj. - ocena projektu wraz z dokumentacją

Przedsięwzięcie

Temat: System zarządzania zadaniami

Technologie:

- ▶ JavaFX - IntelliJ
- ▶ MySQL (MariaDB)
- ▶ GitLab
- ▶ Jira

Efekt po 6 **Sprintach**:

- ▶ gotowy produkt
- ▶ instalator
- ▶ instrukcja obsługi, dokumentacja, UML

Zasady SCRUM

- ▶ Samoorganizacja wewnątrz zespołu 5 os.
- ▶ Odpowiedzialność zbiorowa w ramach grupy
- ▶ 2 tygodniowe **Sprinty**
- ▶ Utrzymywanie aktualnych diagramów Jira:
 - ▶ przesuwanie **Tasków** przez osoby za nie odpowiedzialne
 - ▶ obowiązkowy **Daily Sprint** w środku tygodnia pomiędzy zajęciami
 - ▶ razie potrzeby komentarze, etykiety

Zasady SCRUM

- ▶ Każdy Sprint ma być działającą nową wersją spełniającą przede wszystkim **Cel Sprintu**, ale i jego wszystkie **Story**
- ▶ Możliwość nie wypełnienia **tylko jednego celu Sprintu** z wszystkich 6!
- ▶ Zmiany w **Product Backlogu** robimy tylko w uzgodnieniu z **Product Ownerem** (prowadzącym)
- ▶ Pozostałe tablice są do dyspozycji zespołu:
Sprint Backlog, In process, Ready for Testing, Testing, Done

Daily Sprint

- ▶ Krótkie udokumentowane spotkanie wszystkich członków
 - ▶ na żywo na Teams
 - ▶ minimum 3 razy na **Sprint**
 - ▶ nie może to być offline - wszyscy mają być obecni!
- ▶ Każdy odpowiada bardzo zwięźle na 3 pytania:
 - ▶ Co zrobiłem wczoraj, co pomogło Zespołowi Deweloperskiemu przybliżyć się do osiągnięcia **Celu Sprintu**?
 - ▶ Co zrobię dzisiaj, co pomoże Zespołowi Deweloperskiemu przybliżyć się do osiągnięcia **Celu Sprintu**?
 - ▶ Czy widzę jakiegokolwiek przeszkody mogące uniemożliwić mi lub Zespołowi Deweloperskiemu osiągnięcie **Celu Sprintu**?

Definition of Done

- ▶ **task** ma być działający tak, aby pozwalał spełnić **cel sprintu**
- ▶ kod musi być:
 - ▶ udokumentowany (dokumentowany) - javadoc
 - ▶ zgodny ze standardami kodowania
 - ▶ przetestowany - JUnit !!!!!
- ▶ tylko **task** zgodny z **Definition of Done** może być przesunięty do **Done**

Definition of Done wyjaśnienia

- ▶ jeżeli tematu nie wyjaśniono wcześniej kod musi być:
 - ▶ okomentowany/udokumentowany:
 - * nieznanomość javadoc nie zwalnia z komentowania kodu stosownie do aktualnego stanu wiedzy zespołu
 - ▶ zgodny ze standardami kodowania:
 - * nieznanomość standardów kodowania nie zwalnia ze uporządkowania i schludności (NetBeans format, nazewnictwo klas i zmiennych, itd.)
 - ▶ przetestowany:
 - * JUnit
 - * ręcznie przez inną osobę niż autor, co musi być widoczne w Jira

Podział na 3 grupy

Przydział referatów

Przygotowanie na kolejne zajęcia

- ▶ Kolokwium z podstaw Javy:
 - * zakres: roz. 3-10, 11-13 bez metod, 14, 17, 19-27

https://mjochab.edu.pl/Programowanie_zespolowe/java_tutorial.pdf

- * Bruce Eckel „Thinking in Java” - edycja polska

- ▶ Dopracowanie specyfikacji wymagań i pomysłu na aplikację
- ▶ Założenie konta na GitLabie

Specyfikacja systemu obsługi ISP:

- ▶ Aplikacja powinna mieć kilka modułów:
 - ▶ administracja użytkownikami (role),
 - ▶ administracja klientami,
 - ▶ moduł generowania pakietów usług,
 - ▶ moduł konfiguracji
- ▶ odpowiednie funkcjonalności mają być przypisane do odpowiednich grup pracowników, np.:
 - ▶ księgowi na dostęp danych o opłaceniu rachunków przez klientów
 - ▶ dział marketingu może tworzyć nowe oferty i pakiety wraz z cenami
 - ▶ handlowcy mogą dodawać nowe umowy i klientów
 - ▶ raporty w PDF, itp.

Specyfikacja systemu zarządzania projektami:

- ▶ Aplikacja powinna mieć kilka modułów: administracja użytkownikami (role), moduł raportów, moduł konfiguracji
- ▶ Użytkownik zwykły widzi swoje zadania
- ▶ Administrator zarządza wszystkim
- ▶ Kierownik widzi zadania swojej grupy
- ▶ Raporty w pdf

Cele i zakres systemu

- ▶ dokładne zdefiniowanie problematyki projektu, czyli do czego ma służyć tworzony system komputerowy i jakie problemy mają być rozwiązywane za jego pomocą,
- ▶ ustalenie tzw. interesariuszy projektu, czyli osób lub grup osób, które są potencjalnymi użytkownikami wytworzonego oprogramowania,
- ▶ ustalenie jakie są gromadzone dane w organizacji i kto z użytkowników je wprowadza do systemu,

Cele i zakres systemu

- ▶ ustalenie rodzajów potrzebnych raportów (wydruków) które mają być generowane i udostępniane w projekcie oraz kto z użytkowników je generuje i z nich korzysta,
- ▶ ustalenie rodzajów potrzebnych dokumentów które mają być generowane w projekcie oraz kto z użytkowników je ma generować i z nich korzystać,

Cele i zakres systemu

- ▶ ustalenie przepływu informacji w środowisku, w którym ma pracować system,
- ▶ ustalenie uprawnień (praw dostępu) poszczególnych grup użytkowników (np. ustawa o ochronie danych osobowych).

Określenie wymagań

- ▶ **Wymagania funkcjonalne** – dotyczą tego co ma realizować system; jakie ma spełniać funkcje, jakich dostarczać usług, jak zachowywać się w określonych sytuacjach. Wymagania funkcjonalne powinny być kompletne (opisywać wszystkie usługi żądane od systemu) i spójne (nie zawierać stwierdzeń sprzecznych).

Określenie wymagań

- ▶ **Wymagania нефunkcjonalne** – dotyczą tego jak system powinien realizować swoje zadania; np. wymagania dotyczące koniecznych zasobów, ograniczeń czasowych, niezawodności, bezpieczeństwa, przenośności, współpracy z określonymi narzędziami i środowiskami, zgodności z normami i standardami, a także przepisami prawnymi, w tym dotyczącymi tajności i prywatności, itp. Wymagania нефunkcjonalne dla wielu systemów są co najmniej tak ważne jak wymagania funkcjonalne (np. szybkość działania wyszukiwarki może być równie ważna jak precyzja wyszukiwania)