Daniel Górski
Korporacyjna Java
Wykład 13: Podsumowanie pracowni

Kalendarz

- Dzisiaj
  - Podsumowanie pracowni
  - Wstępne oddawanie całych projektów
- 29 maja: zajęcia piątkowe wg kalendarza roku akademickiego
- 5 czerwca
  - Będę dostępny 7:30 – 11:30
  - Możliwość ostatecznej prezentacji projektu i wystawienie ocen

Kalendarz...

- 12 czerwca: Wykład
  - Kilka osób podłączy się do zdalnej konferencji i opowie o swojej ścieżce edukacyjnej i zawodowej
    - Możliwość dyskusji
    - Będą to osoby z 15 – 20 letnim doświadczeniem w IT
- 12 czerwca: Pracownia
  - Będę przeprowadzał testowe rozmowy techniczne
    - Zakres: java, bazy danych, architektura, systemy rozproszone
  - Oceny z pracowni będą już wystawione: na nic to nie będzie rzutować

Podsumowanie pracowni

Praca korporacyjnego programisty

- Oczywiście, że zdarzają się podczas pracy zawodonej w korporacji:
  - Rozpoczęcie pisania całym zespołem nowej aplikacji
  - Pisanie ciekawych pilotów: wstępnych wersji aplikacji mających sprawdzić czy coś jest możliwe w realizacji w sensownym koszcie
  - Zdobywanie nowych umiejętności i stosowanie ich w różnych miejscach: zmiana utartych schematów
  - Wspólne zespołowe dowożenie funkcjonalności w dobrej atmosferze

Praca korporacyjnego programisty...

- Jednak często się zdarza:
  - Żmudne poszukiwania błędu: często nieswojego, a wręcz powstałego nie z powodu błędu jednej osoby
  - Problemy z pracą w wiele osób nad jedną rzeczą:
    - Często nie można wystarczająco zrównoleglić pracy
    - Regularnie czyjaś praca nie wchodzi do finalnej wersji / jest usuwana / trzeba ją powtórzyć aby dodać do nowej wersji...
  - Część kodu / funkcjonalności z którymi się zżyliśmy wylatuje z projektu
  - Wyobrażamy sobie nowe ciekawe funkcjonalności, a mamy pisać żmudnie nudny powtarzalny kod...

Praca korporacyjnego programisty...

- Sytuacja idealna typu: dostajemy od razu kompletne wymagania, które mamy zaimplementować jest sytuacją teoretyczną :-)
- Zdarzają się projekty, które ewoluują przewidywalnie, ale na przestrzeni wielu lat przechodzą często znaczną metamorfozę
- Z perspektywy korporacji ważne jest żeby mieć:
  - Ustandaryzowany kod
  - Wymiennych pracowników
  - Poza pewnymi wąskimi gardłami ważniejsze jest aby mieć kod bardziej czytelny kosztem wydajności

Praca korporacyjnego programisty...

- A z perspektywy programisty trzeba wyważyć...
  - Jeśli jest się niezastąpionym w jakimś obszarze to jest się trudno wymienialnym, ale z drugiej strony ciężko się odkleić od tego obszaru
  - Jeśli wiedza w zespole jest dobrze rozpropagowana i nie ma "silosów" to jest się łatwo wymienialnym zasobem

Zakres pracowni

- Plan przedmiotu i pracowni opracowałem samodzielnie
- Przyświecała mi generalnie myśl aby stworzyć przedmiot jaki przygotowałby mnie do pracy programisty na etat w korporacji przed jej rozpoczęciem
- Wymyśliłem, że priorytetem będzie praca zespołowa nad jedną rozrastającą się bazą kodu przed cały semestr

Zakres pracowni

- Większy nacisk położyłem na to, żeby baza kodu była większa niż trudna
- Przy okazji starałem się nakierowywać Was na to jak można pisać generyczny kod i cieszy mnie fakt, że pod koniec wszystkie zespoły dosyć stabilne silniki, które były dobrze rozszerzalne [S**O**LID]
- Ponieważ mam już za sobą wiele lat w branży to na niektóre rzeczy zwracałem mocno uwagę, bo czas przedmiotu byłby za krótki abyście mieli okazję porządnie się na nich potknąć

Punktacja

- 90 punktów z cotygodniowych prezentacji
- 60 punktów z finalnej oceny projektu:
  - 20p: wybrane ścieżki poprawnego działania
  - 10p: wybrane ścieżki błędnego działania
  - 20p: architektura aplikacji
  - 10p: testy
  - 0p: clean coding (punktacja od -7p do 0p)

Wybrane ścieżki poprawnego działania

- 20p: wybrane ścieżki poprawnego działania
  - 17p: poprawne obliczanie figur
  - 1p: zapis do pliku
  - 1p: sortowanie figur
  - 1p: usuwanie figury

Poprawne obliczanie figur

- 17p: poprawne obliczanie figur
  - 1p: Kradrat
  - 1p: Koło
  - 1p: Prostokąt
  - 2p: Romb
  - 1p: Trójkąt równoboczny
  - 2p: Trójkąt równoramienny
  - 2p: Trójkąt prostokątny
  - 1p: Trójkąt dowolny
  - 4p: Trapez równoramienny
  - 1p: Elipsa
  - 1p: Sześciokąt foremny

Wybrane ścieżki błędnego działania

- 10p: wybrane ścieżki błędnego działania
  - 3p: niepoprawne wejście
    - 1p: wprowadzenie nie liczby, ujemnej liczby
    - 1p: niemożliwy trójkąt
    - 1p: niemożliwy trapez
  - 7p: zdublowane figury
    - 1p: wprowadzenie takiego samego trójkąta (inna kolejność boków)
    - 1p: wprowadzenie prostokąta takiego jak istniejący kwadrat
    - 1p: wprowadzenie elipsy o dwóch półosiach równych i takich samych jak istniejące koło
    - 1p: wprowadzenie koła i koła o dwukrotnie mniejszym promieniu, dwukrotne podwojenie
    - 1p: wprowadzenie koła o średnicy 13, wprowadzenie trójkąta prostokątnego (5, 12, 13), próba opisania koła na tym trójkącie
    - 1p: wprowadzenie trapezu jako istniejący prostokąt
    - 1p: wprowadzenie trójkąta dowolnego jak istniejący równoramienny

Architektura aplikacji

- 20p: architektura aplikacji
  - 5p: Dodanie nowej figury: co trzeba zaimplementować, gdzie dodać zmiany
  - 5p: Dodanie nowej wersji językowej: ile trzeba wykonać nowej pracy poza samym tłumaczeniem (0p jeśli w tej chwili nie ma dwóch wersji językowych)
  - 3p: Dodanie nowej akcji: usunąć wszystkie figury wybranego typu: gdzie trzeba dodać zmiany, czy trzeba dodać ręcznie listę typów figur
  - 3p: Obsługa błędów, dedykowane wyjątki
  - 2p: Dodanie nowego sposobu sortowania: tylko po części "za przecinkiem" pola powierzchni: gdzie trzeba dodać zmiany
  - 2p: Gdzie jest informacja o tym jakie właściwości ma dana figura i ile właściwości jest potrzebne do wyliczenia

Testy

- 10p: testy
  - 1p: wszystkie testy przechodzą
  - 9p: ręczna zmiana w kodzie i odpalenie testów:
    - 1p: Źle liczony bok kwadratu
    - 1p: Źle liczone pole powierzchni prostokąta
    - 1p: Źle liczony obwód koła
    - 1p: Źle liczone podwojenie trójkąta dowolnego
    - 1p: Źle liczona wysokość w trójkącie równoramiennym
    - 1p: Źle liczona wysokość trapezu
    - 1p: Źle liczone pole powierzchni trójkata prostokątnego
    - 1p: Źle liczone pole powierzchni w sześciokącie
    - 1p: Źle liczona przekątna rombu

Clen Coding

- 0p: clean coding
  - Można stracić punkty za:
    - 1p: niejednolite formatowanie / ekstremalnie długie linie
    - 1p: nazewnictwo klas / metod / zmiennych nie wg standardowych konwencji, z błędami
    - 1p: konstrukcje niemile widziane w programowaniu obiektowym: refleksje / rzutowanie klas
    - 3p: bardzo długie metody, dublowanie kodu, zakomentowany kod
    - 1p: nieintuicyjne pakietowanie
  - Dotyczy kodu, nie dotyczy testów

Progi

- 5: 140p
- 4+: 130p
- 4: 120p
- 3+: 110p
- 3: 100p
- Zespół Ich Troje, który skończył jako zespół 2-osobowy będzie oceniony indywidualnie