Dokumentacja końcowa

Projekt grupowy PROI - Lato 2021

Przedmiot: Programowanie Obiektowe
Temat projektu: Firma testująca gry
Autorzy: Jan Jędrzejewski, Szymon Wysocki, Patrycja Wysocka
Data: 10/06/2021 r.

1. Założenia projektu

I. W firmie pracuje n testerów gier (ich liczbę określa użytkownik).
II. Firma udostępnia możliwość testowania gier w ramach określonych specjalizacji (związanych z typami gier).
III. W firmie pracuje manager zajmujący się przyjmowaniem ofert od klientów, którymi są producenci gier oraz przydzielaniem gier do konkretnych testerów.
IV. Zgłaszane do testowania gry są przydzielane do pewnych stałych kategorii, od których zależy pula potencjalnych testerów.
V. Podczas zgłaszania gier w celu ich testowania producent uzyskuje cennik w zależności od ilości godzin testowania oraz ilości kompetentnych testerów.
VI. Po zakończeniu testowania danej gry producent gry otrzymuje raport z wynikiem testowania.
VII. Firma notuje terminowość opłaty za testowanie gier. Producent gier obowiązuje się do zapłaty w przeciągu 8 h (jeden dzień roboczy). W przeciwnym razie, za każdą 1 h zwłoki obowiązuje dopłata 15 zł.

2. Hierarchia klas

Employee – reprezentuje pracownika. Każdy z nich ma swój identyfikator oraz stawkę. Bonus() to funkcja wirtualna.

Tester – dziedziczy po Employee, dodatkowo istnieje możliwość sprawdzenia umiejętności testowania danego typu gry, ponieważ każdy tester posiada swój zestaw typów które jest w stanie testować.

Manager – dziedziczy po Employee, kontroluje firmę, ma swój zespół testerów, którymi rozporządza: przydziela obowiązki, wypłaca pieniądze, pilnuje terminowości zapłat za zlecenia. Zawiera zbiór wszystkich zleceń.

ReviewRequest – reprezentacja zlecenia, zawiera wszystkie najważniejsze informacje – id, typ i tytuł gry, czas realizacji, informację o płatności w terminie, cenę, ocenę gry.

Game – reprezentuje grę, posiada swój identyfikator, typ, tytuł, wydawcę, a później także i oceny (wystawiane w trakcje testowania gry).

Publisher – posiada swój zestaw gier, zgłasza zlecenia aby przetestować swoje gry.

Simulation – bazując na powyższych klasach tworzy symulację, generuje testerów, gry oraz wydawców (z plików) a także managera. Jest odpowiedzialna za upływ czasu, wypisywanie stanu symulacji, oraz zapis do pliku.

3. Podział na pliki

Wyżej wymienione klasy zostały zaimplementowane w odpowiadających im plikach:

employee.h oraz employee.cpp - klasy Employee, Tester i Manager
game.g oraz game.cpp - klasa Game i ReviewReques
publisher.h oraz publisher.cpp - klasa publisher
simulation.h oraz simulation.cpp - klasa Simulation

main.cpp - plik gotowy do uruchomienia, tworzy obiekt klasy symulacji i ją przeprowadza.

W folderze projektu znajdują się także pliki tekstowe game_names.txt i publisher_names.txt, które przechowują nazwy gier oraz ich deweloperów na podstawie których są generowane obiekty klas Game i Publisher. Do pliku save.txt zostaje zapisany przebieg symulacji.

4. Uruchomienie symulacji

Program należy wywołać wedle poniższego wzoru z linii polecenia:

- (odpowiednia kompilacja np):

  g++ *.cpp -o main

- main.exe {ilość godzin trwania symulacji} {ilość testerów} {liczba wydawców} {liczba gier}

Program jest zabezpieczony przed wprowadzaniem niepoprawnych danych. Cały przebieg symulacji zostaje wyświetlony w terminalu, a kopia wyświetlonych informacji zostaje zapisana do pliku save.txt.

5. Sposób przeprowadzenia symulacji

Po wprowadzeniu danych przy uruchamianiu symulacji zostaną wygenerowane klasy:

Managera razem z wprowadzoną liczbą dostępnych testerów, którym przypisujemy kilka wylosowanych gatunków gier w jakim się specjalizują, oraz wylosowaną stawkę godzinową.

Lista Publisherów, o długości uprzednio wprowadzonej, którym wybieramy nazwę i przypisujemy jednakową liczbę gier, dla których losujemy gatunek i wybieramy nazwę z pliku.

Następnie startuje symulacja godzina po godzinie dla liczby wprowadzonych godzin. Co godzinę jest szansa 20% że zostanie złożone zlecenie testowania gry. Wówczas wybierany jest deweloper, a następnie wybierana jest jedna z jego gier dla której składamy zlecenie, na wylosowaną liczbę godzin (od 10 do 20 godzin).

Co godzinę symulacja wyświetla komunikaty zwrócone przez managera. Informujące o przyjętych zleceniach, statusie aktualnie testowanych gier, zleceniach oczekujących na opłatę. Raz na tydzień, czyli co 40 godzin, manager wypłaca swoim pracownikom pensję.

Po zakończeniu symulacji zostaje wyświetlone podsumowanie, w którym znajdziemy zestawienie wykonanych zleceń oraz informacje finansowe, ile zarobił manager a ile firma.

6. Wskazanie wykorzystywanych mechanizmów

W projekcie zostały wykorzystane poniższe elementy biblioteki STL:

- std::vector - m.in. symulacja posiada vector wydawców, a oni vector gier
- std::list - w klasie Manager
- std::set - do wyrażania gatunków gier reprezentowanych jako enum
- std::stack - w klasie Simulation
- std::shared_ptr - m.in. w klasie Manager
- std::string - na przestrzeni całego projektu

Korzystanie z liczb pseudolosowych:

W klasie Simulation wykorzystano funkcję generator() z biblioteki <random> generującą liczby z podanego zakresu. Na ich podstawie składamy zlecenia, wybieramy ile godzin należy testować grę oraz przypisujemy testerom ich pensje.

7. Opis zidentyfikowanych sytuacji wyjątkowych i ich obsługi

W programie zawarto obsługę wyjątków wynikającą z podania niepoprawnych argumentów przy uruchamianiu programu. Obsłużono podawanie zbyt duże jak i zbyt małe wartości, oraz nieodpowiednią liczbę argumentów poprzez wypisywanie komunikatów w konsoli.

8. Repozytorium

Poniżej znajduje się link do repozytorium na wydziałowym gitlabie, w którym znajduje się implementacja projektu, oraz niezbędne pliki do uruchomienia.

https://gitlab-stud.elka.pw.edu.pl/swysocki/proi_21l_projekt_jj_pw_sw