Testovacie scenáre

Debug mód pre vývojára

1. Test spustenia animácie

    a. používateľ spustí animáciu napísaním OAL kódu alebo jeho vygenerovaním
    b. OAL kód následne uloží v ľubovoľne vybranom adresári
    c. vloží vytvorený súbor pomocou tlačidla open
    d. kliknutím tlačidla PLAY sa animácia začne vykonávať
    e. butony s možnosťami rozloženia sa presunú na hornú lištu
    f. na ich mieste je okno s OAL kódom

2. Test vytvorenia skriptu

    a. skript sa dá vytvoriť pomocou tlačidla CREATE
    b. OAL kód užívateľ napíše do prázdneho modulu XUML alebo vygeneruje vyklikaním z ponuky pomocou modulu search
    c. použitím tlačidla check sa skontroluje kód

3. Test uloženia skriptu

    > 2. c.

    a. po vytvoreni skriptu sa súbor (.txt) uloží do vybraného adresára, ktorý si vyberá používateľ (tlačidlo SAVE)
    b. súbor zostáva uložený, pokiaľ ho niekto fyzicky nezmaže

4. Test behu animácie

    > 1. d.

    a. v okne s napísaným/vygenerovaným skriptom (XUML)sa časovo synchronizovane s animáciou vyfarbuje OAL kód podľa návrhu používateľského rozhrania
        i. každý jeden príkaz (call) je zvýraznený práve vtedy, keď je jeho reprezentácia zvýrazňovaná na diagrame
        ii. po skončení vysvietenia príkazu na diagrame sa príkaz v skripte (XML) vráti do pôvodného stavu
    b. po skončení animácie OAL kód ostane v pôvodnom stave

5. Test korektného načítania súboru

    > 1. c

    a. užívateľ vyberie súbor pomocou tlačidla OPEN, pokiaľ žiaden neexistuje - (>2. a. - 3.b.<)
    b. otvorí sa vyhľadávací modul Unity, kde korektne uložený súbor (> 2. a. - 3.b.<) užívateľ nájde
    c. vznikne nový modul (XUML) s textom s načítaného súboru
    d. text v XML module nie je rozdielny od uloženého textu v textovom súbore
    e. text je rovnako formátovaný ako text v uloženom textovom súbore

6. Test prekrývania

    >4. a.

    a. vzniknutý modul XUML neprekrýva žiaden už existujúci modul rozhrania
    b. presunutý modul s možnosťami rozloženia neprekrýva žiaden existujúci modul rozhrania
    c. žiaden modul rozhrania neprekrýva diagram v používateľskom rozhraní

7. Test čitateľnosti

    >1. d.

    a. vysvecovaný riadok v OAL skripte je vysvecovaný odlišnou farbou ako pôvodne vypísaný riadok
    b. po skončení vysvietenia riadku sa farba riadku opäť zmení na pôvodnú

8. Test korektného skončenia animácie

    >4. b.

    a. po kliknutí na krížik tlačidla PLAY s rozhrania zmizne modul (XML) s OAL kódom
    b. modul s možnosťami rozloženia sa vráti na pôvodné miesto z pred spustenia animácie (< 1. d.)

9. Test korektného OAL skriptu

    >4. a..

    a. skript spĺňa syntax jazyka OAL
    b. každý príkaz skriptu musí spĺňať formát jazyka OAL
    c. názvy metód a tried musia byť obsiahnuté v načítanom diagrame
    vzor:

        call from Client::PrepareVisitors() to CreditCard::accept() across R6;
        call from GoldCreditCard::getGoldCardValue() to OfferVisitor::visitGoldCreditCard() across R11;

call from GasOfferVisitor::visitSilverCreditCard() to
BronzeCreditCard::getBronzeCardValue() across R3;

10. Test nesprávneho OAL skriptu
    a. po zásahu do korektne uloženého OAL skriptu a následnom spustení animácie animácia spadne na chybe
    vzor:

    “**coment**”
    “**random text**” call from Client::PrepareVisitors() to
    CreditCard::accept() across R6;
    2.
    call from GoldCreditCard::getGoldCardValue() to
    OfferVisitor::**nonexistingmethod**() across R11;
    3.
    neprimerane veľký počet volaní

Debug mód pre používateľa

Implementácia tiel metód

1. Test formátu tela metód
    a. Ak metóda má zadefinované telo
        i. jej telo je zobrazené v samostatnom module
        ii. label modulu je názvom danej metódy
    b. Ak metóda nemá zadefinované telo
        i. telo modulu je prázdne
        ii. label modulu je názvom danej metódy

2. Test pridania tela metódy
    a. kliknutím na metódu v diagrame sa otvorí plávajúci modul
    b. ak metóda nemá uložené telo, otvorí sa prázdne editovacie okno (modul)
    c. ak už existuje telo metódy, otvorí sa editovacie okno s jeho obsahom
    d. v tele modulu môžeme editovať resp. pridať telo metódy

3. Test uloženia tela metódy
   -2.d.
    a. ak je dokončené editovanie, použijeme tlačidlo SAVE
    b. ak je telo pridávané prvýkrát, telo metódy sa uloží do vybraného súboru (.oal), v adresári projektu
    c. ak je telo upravované, uloží sa do pôvodného, už vytvoreného súboru s upravenými zmenami
4. Test behu animácie
    a. v okne s napísaným/vygenerovaným skriptom (XUML)sa časovo synchronizovane s animáciou vyfarbuje OAL kód podľa návrhu používateľského rozhrania
    b. v module pribudne rozbaľovacia ponuka s všetkými metódami vyskytujúcimi sa v animácii
    c. po kliknutí na niektorú z metód sa otvorí nové plávajúce okno
        i. ak metóda má definované telo, zobrazí sa v plávajúcom module, label modulu je názov metódy
        ii. ak metóda telo definované nemá, zobrazí sa modul s lavel názvom metódy a prázdnym telom
    b. po skončení animácie OAL kód ostane v pôvodnom stave, otvorené metódy zostanú otvorené
5. Test formátu tela metódy
    a. telo metódy spĺňa syntax jazyka OAL
    b. každé volanie v tele metódy musí smerovať z danej metódy
6. Test korektného tela metódy
    a. `Názov/label: Game::CreateArmy()`
    b. `obsah:`

```
call from Game::CreateArmy() to
AbstractFactory::CreateWarrior() across R4;
call from Game::CreateArmy() to
AbstractFactory::CreateWarrior() across R4;
call from Game::CreateArmy() to
AbstractFactory::CreateWarrior() across R4;
call from Game::CreateArmy() to
AbstractFactory::CreateRanger() across R4;
call from Game::CreateArmy() to
AbstractFactory::CreateRanger() across R4;
call from Game::CreateArmy() to
AbstractFactory::CreateMage() across R4;
```

7. Test nesprávneho formátu tela metódy
   a. názov/label:
      ```
      AbstractFactory::CreateRanger()
      ```
   b. telo:
      ```
      call from Game::CreateMage() to TrollFactory::CreateRanger()
      across R2;
      ```