Házi feladat munkanapló: asd

Integrációs és ellenőrzési technikák (VIMIAC04)

Csapattag 1 (Bakurecz Ádám, LOT078, adambakurecz)

Részfeladat 1 (összesen kb. 7 óra):

* Unit teszteket írtam a MechanicCharacter és a SaboteurCharacter osztályokhoz
* A MechanicCharacter tesztelése során különböző konstruktorokat ellenőriztem, hogy helyesen állítsák be az osztály attribútumait. Ezen kívül teszteltem a metódusokat, mint például a csövek és pumpák javítása, felvétele és lerakása. Az ilyen tesztek segítenek biztosítani, hogy a mechanikus karakter a játék különböző helyzeteiben is megfelelően működjön, és helyesen kezelje azokat az eseteket is, amikor nincs pumpa vagy cső a karakter birtokában.
* A SaboteurCharacter tesztelése során szintén figyelembe vettem a konstruktorokat és a főbb funkciókat, mint például a csövek csúszóssá tétele. Ezek a tesztek biztosítják, hogy a szabotőr karakter megfelelően végre tudja hajtani a neki szánt romboló tevékenységeket a játékban.
* A tesztek írása során a Mockito-t használtam a függőségek mockolásához. Ez lehetővé tette az izolált tesztelést, ahol csak a tesztelt osztály viselkedésére koncentráltam.
* Összességében ezek az egységtesztek jelentősen növelik a kódbázis megbízhatóságát és karbantarthatóságát, hiszen a jövőbeli fejlesztések során könnyen ellenőrizhetővé válik, hogy a változtatások nem okoznak-e hibát a meglévő funkciókban.

Részfeladat 2 (összesen kb. 2 óra):

* Jacoco teszt lefedetteség mérő eszköz beüzemelése

Részfeladat 3 (összesen kb. 3 óra): Pull Request reviews

Reviewed PRs:

* Static analysis #2
* Maven setup #4
* Create maven.yml #7
* UI tesztek Tosca-val #10

Csapattag 2 (Gál Kincső, BBFQ1I, galkincso)

Részfeladat 1 (összesen kb. 10 óra):

* Feladatcsoport: Technológia fókusz
* Konkrét feladat: Build keretrendszer beüzemelése, ha még nincs (Maven, Gradle...) + CI beüzemelése, ha még nincs (GitHub Actions, AppVeyor, Azure Pipelines...)
* Első feladatrész: Build keretrendszer beüzemelése, amihez a Maven keretrendszert választottam
* A lehetséges keretrendszer opciók tanulmányozása után arra jutottam, hogy a projekt tekintetében a Maven keretrendszer lenne a megfelelőbb.
* Egyszerű a beüzemelése, könnyen tanulható, előnyösebb kisebb projektek esetén, könnyen átlátható mappastruktúrát biztosít, stb...
* A Maven keretrendszer IntelliJ segítségével beállítható volt., dependecie- vel nekem nem volt szükséges kiegészítenem.
* Második feladatrész: CI beüzemelése GitHub Actions segítségével
* A feladat megoldásához tutorial videókat néztem, összehasonlítva a megvalósítási folyamatokat, illetve a megvalósított funkciókat is.
* Volt, ahol a GitHub Actions ajánlotta CI workflowt valósították meg, illetve volt, ahol saját magunknak készíthettük el.
* Én az előbbit választottam, mivel pontosan volt nekem megfelelő workflow, azaz Java + Maven
* A Dependency graph részt kitöröltem, illetve egy Checkouttal bővítettem az alap vázat
* A célom a projekt buildelése, hogy buildelhető és futtatható küd legyen mindig feltöltve a GitHub- ra

Részfeladat 2 (összesen kb. 1 óra):

* Review requestek:
* UI Tesztek Tosca- val:
* Ezt a feladatrészt mások is ellenőrizték, az ők ellenőrzésükkel én is egyetértettem, nekem is jónak tűntek az elvégzett feladatok
* Unit tests
* A projekt egyik készítőjeként könnyebben sikerült megértenem a megírt tesztek működését.
* Szépen megírt, jól strukturált tesztek.

Csapattag 3 (Ignáth Dávid, C0VCBW, saviourdavid)

Részfeladat 1 (összesen kb. 15 óra):

* Feladatcsoport: Termék/felhasználói fókusz
* Konkrét feladat: UI tesztek készítése (Selenium, Tosca, Appium...)
* A UI teszteket a **Tricentis** vállalat **Tosca 16.0 LTS** verziójú applikációval valósítottuk meg együtt Romoda Bálinttal.
* Ehhez először is licenszt kellet igényelni, majd letölteni a megfelelő repositoryt a **Tricentis** weboldaláról. Sikeres telepítés után a feladatokat több részre is osztottuk. A tesztelés során arra törekedtünk, hogy olyan hibákat térképezzünk fel, javítsunk ki, amelyeket kijavítva minden felhasználónak minél pozitívabb legyen a játékélménye. Bálint által készített teszteket leellenőriztem, esetleg kijavítottam, mindketten arra törekedtünk, minél egységesebbek legyenek a tesztek felépítése.
* A **Toscan Commander Repository file** létrehoztam a **"UITestingToscan" mappán belül „iet\_teszt néven**”. Két nagyobb mappába soroltam a teszteket: PreGameTest, PostGameTests. A PostGameTests-en belül minden egyes játékon belüli parancsra készült egy-egy mappa több testcase-el. Legtöbb Test Case-t Manual Test Case-ként készítettem el, van amelyikhez tartoznak képernyőképek is a könnyebb megértéshez.
* **Általám készültek az alábbi tesztek**:
* PreGameTest: kipróbáljuk az összes UI elementet a játék előtti képernyőn.
* Attach Pipe tests: játékon belül az „a” gomb megnyomására több testcase
* Movement tests: játékon belül az „m” gomb megnyomására több testcase
* Pick up new pipe tests: játékon belül az „n” gomb megnyomására több testcase
* Pick up pipe tests: játékon belül az „p” gomb megnyomására több testcase
* Pick up pump tests: játékon belül az „b” gomb megnyomására több testcase
* Plcae pump tests: játékon belül az „q” gomb megnyomására több testcase

Részfeladat 2 (összesen kb. 5 óra):

* Feladatcsoport: Termék/felhasználói fókusz
* Konkrét feladat: Manuális tesztek megtervezése, végrehajtása és dokumentálása vagy exploratory testing
* Választott feladatrész: **Exploratory testing.**
* A feladatot együtt Romoda Bálintal csináltuk meg. Először arra törekedtem, hogy megértsem, milyen megtaláljam az összes lehetséges módszert, a játékkal való interaktálásához. Ehhez nem kellett a játék szabályait elolvassam, mivel én is megcsináltam ugyanezt a feladatot 1 évvel ezelőtt, csak másik csapattal.
* Legjobb módszernek találtam a csapat által készített dokumentáció elolvasását. Ezután könnyen el tudtam indítani a játékot (Visual Studioval) és elkezdtem potenciális hibákat keresni. A dokumentációhoz egy sablont készítettem el, amit minden egyes dokumentumra felhasználtam.
* A teszteket egy meghatárott időpontban egy adott egységre fókuszálva végeztem el, folyamatosan dokumentálva, milyen scenario-kat viszgaltam meg, milyen lépéseket végeztem el. Ha találtam potenciális hibákat azokat feljegyeztem. Miután végeztem egy tesztel konzultáltam Bálinttal, aki esetleg belejavított a dokumentációba vagy felvetett egy olyan esetet, amit én nem vizsgáltam meg. Ha esetleges hibákat találtam, azokat jelentettem az egész csapatnak és megbeszéltük mi a teendő.
* A dokumentáció egyes exploratory teszre az „ExploratoryTests” mappában találhatóak meg. **Az alábbiakat végeztem el:**
* NewPumpsAndPipesTest
* PregameTest
* WaterflowTest
* WinningTest

Részfeladat 3 (összesen kb. 30 perc):

* Feladatcsoport: **Review Pull Request**
* Konkrét Pull Request az alábbi feladatra: Manuális kód átvizsgálás elvégzése az alkalmazás egy részére (GitHub, Gerrit...) + Statikus analízis eszköz futtatása és jelzett hibák átnézése (SonarCloud, SpotBugs, VS Code Analyzer, Codacy, Coverity Scan...).
* Szilágyi Anna által készített kód átvizsgálást vizsgáltam meg. Vizsgálás során jól meg tudtam ismerkedni a kód szerkezetével, és potenciális hibákat is kerestem. Mivel nem találtam hibát hamar approve-oltam.

Csapattag 3 (Romoda Bálint, S3BYQU, balu0105)

Részfeladat 1 (összesen kb. 12 óra):

* Feladatcsoport: Termék/felhasználói fókusz
* Konkrét feladat: UI tesztek készítése (Selenium, Tosca, Appium...)
* A UI teszteket a **Tricentis** vállalat **Tosca 16.0 LTS** verziójú applikációval valósítottuk meg együtt Ignáth Dáviddal.
* Ehhez kérni kellet egy licencet, majd letölteni a megfelelő repositoryt a **Tricentis** weboldaláról. Sikeres telepítés után a feladatokat több részre osztottuk, főleg a tesztelendő programban található parancsok alapján. A tesztelés során arra törekedtünk, hogy olyan hibákat térképezzünk fel, javítsunk ki, amelyeket kijavítva minden felhasználónak minél pozitívabb legyen a játékélménye. A Dávid által készített teszteket átnéztem
* **Általám készített tesztek:**
* Fix Pipe tests: játékon belül az ,,f" gomb megnyomására több testcase
* Holey Pipe Test: játékon belül a ,,h" gomb megnyomására több testcase
* Input/output: játékon belül az ,,i" és ,,o" gombok megnyomására több testcase
* Skip tests: játékon belül az ,,x" gomb megnyomására több testcase
* Sticky tests: játékon belül a ,,t" gomb megnyomására több testcase

Részfeladat 2 (összesen kb. 5 óra):

* Feladatcsoport: Termék/felhasználói fókusz
* Konkrét feladat: Manuális tesztek megtervezése, végrehajtása és dokumentálása vagy exploratory testing
* Választott feladatrész: **Exploratory testing.**
* A feladatot együtt Ignáth Dáviddal csináltuk meg. Az előző feladatrészemben már egészen megismerkedtem a játék működésével, így arra nem kellett már koncentrálnom.
* A célom az volt, hogy a játékot minél furcsább módon játszam, ennek köszönhetően a játék alapjait is tudtam tesztelni, azonban az általam előidézett speciálisabb eseteket is meg tudtam figyelni.
* A tesztelés során feljegyeztem a játékkal való összes interakciómat, és azt, hogy mire számítok a végrehajtásukkor. Amennyiben különös működést fedeztem fel, azt feljegyeztem, azonban a játék általában megfelelően működött. A hiba könnyebb megértése érdekében bizonyos helyeken elhelyeztem képeket.
* A dokumentáció egyes exploratory teszre az „ExploratoryTests” mappában találhatóak meg. **Az alábbiakat végeztem el:**
* MovementWithPipePickedUpTestDocumentation
* MultipleStickyPipeTestDocumentation
* NonMovementPickedUpPipeTestDocumentation
* SpecialPickedUPipeTestDocumentation

Csapattag 5 (Szilágyi Anna, I7F4U6, szannaa)

Manuális kód átvizsgálás (összesen kb. 4 óra):

* IntelliJben megnyitottam a projektet és ott ellenőriztem azokat az osztályokat, amiket én írtam még a projekt laborban.
* Észrevételek
* Helykitöltő metódusokat meg lehetne jelölni absztraktként és az alosztályokban megvalósítani őket
* toString metódus lehetne olvashatóbb
* Elírások
* Az objektumorientált elveket megszegő függvények (nem javítottuk, mert ez a rossz tervezés eredménye volt és ezek a működést nem befolyásolják).

Statikus analízis eszköz futtatása és jelzett hibák átnézése (összesen kb. 7 óra):

* A statikus kódanalízis során SonarLint és SonarCloud segítségével számos hibát azonosítottam, különösen az elnevezési konvenciók terén, ezeket nem javítottam, mert sok volt belőlük és nem tanultam volna sokat belőle.
* A random generátorokat nem kellett biztonságossá tenni ebben a projektben, ezt SonarCloudon jeleztem is
* Magas súlyosságú hibákat, például a nem megfelelően használt final típusú adattagok elnevezését, javítottam.
* Továbbá refaktoráltam hosszú és bonyolult függvényeket, elláttam a switch utasításokat default esetekkel, és kiszerveztem a random számgenerátorokat.
* A SonarCloud Analyser még ezek után is hibákkal futott le, de átláthatóságra javult a kódminőség
* Ezek a változtatások hozzájárultak a kód karbantarthatóságának javításához és a potenciális hibák csökkentéséhez.

2024. május. 24.

……………….. ……………….. ………………. …………….. ……………..