# Semestralni prace UPG 2022

*Použil jsem variantu s reálným časem pro kreslení grarfů.*

*Take dokumentace byla rozšířená, přidal jsem do package Datas třídu Inerita, do package Main přidal jswm třídy ChartPanel a Informer, take do třídy DrawingPanel, která se nacházíte v package Main byli přidány veškeré metody. Třída Calculate planets také byla rozšířená, další informace dolů...*

Rozdělil jsem svůj project na 4 package :

## Package Datas.

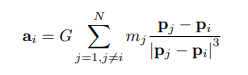
1. Třída se nazývá Acceleration a popisuje zrychlení objektu, zrychlení rozdělil jsem na 2 složky ( zrychlení po X a zrychlení po Y), všechny atributy jsou privátní a přístup udělán přes getry a setry také existuje bezparametricky konstruktor pro zrychlení, který nastavuje všechny hodnoty na 0.
2. Třída se nazývá Position a popisuje pozici objektu ve vesmíru, rozdělil jsem pozice taky na na 2 složky ( pozice po X a pozice po Y), všechny atributy jsou privátní a přístup udělán přes getry a setry také existuje bezparametricky konstruktor pro pozici, který nastavuje všechny hodnoty na 0, taký překryl jsem tam metodu toString pro další výpis informace o pozice planety na obrazovku.
3. Třída se nazývá Reader - vlastní třída pro čtení ze souboru, nevím už proč jsem ji udělal, ale mám tam metodu numString, která předem spočte počet řádek v csv a předá, zatím metoda OpenAndRead udělá pole Stringu a přečte soubor řádku po řádce, nevím proč udělal jsem to tak, šlo by to udělat jednoduše, ale nakopíroval jsem tuto třídu z Ppa2 a použil jsem ji tady.
4. Třída se nazývá Speed a popisuje rychlost objektu, rychlost rozdělil jsem na 2 složky ( rychlost po X a rychlost po Y), všechny atributy jsou privátní a přístup udělán přes getry a setry také existuje bezparametricky konstruktor pro rychlost, který nastavuje všechny hodnoty na 0. taký překryl jsem metodu toString pro další výpis informace o rychlosti planety na obrazovku.
5. Třída se nazývá Weight a popisuje vahu objektu ve vesmíru, všechny atributy jsou privátní a přístup udělán přes getry a setry také existuje bezparametricky konstruktor pro váhu.
6. Třída se nazývá Inertia a popisuje inerci při sloučení dvou planet, také ma v sebe metodu která se nazývá inertionalSum a dělá v podstatě jenom to, že sečtěte dvě různé inercii v jednom na jedné ose

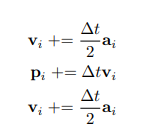
## Package SpaceObjects.

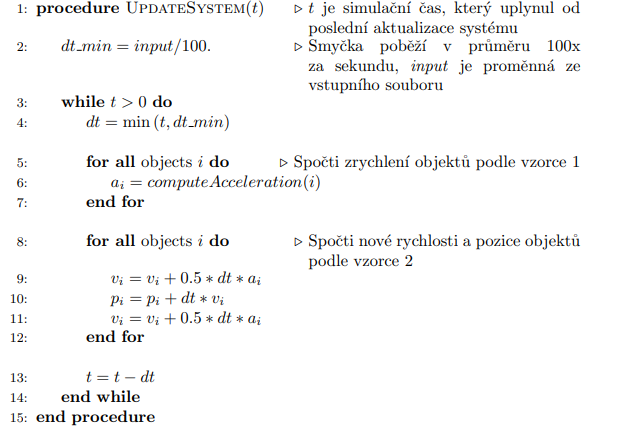
1. Třída se nazývá ObjectSpace a tohle je rodičovská třída pro všechny objekty ve vesmíru - všechny atributy jsou protected pro přístup potomka, taky přidal jsem tam atribut isHit, bolovsky atribut, který ukazuje, zda na planetu někdo kliknul a program musí ji zobrazovat červenou a ukazovat o ní informací. Taký přístup k atributam udělán přes getry a setry a udělán konstruktor pro vytváření vesmírného objektu.
2. Třída se nazývá Comet tohle je třída která dědí od třídy ObjectSpace v podstatě tam přidán jenom atribut rádius, přístup k radiusu udělaný přes getry a setry, konstruktor komety udělán přes super() - dědění od tridy SpaceObject, jenom v konstruktor ještě navíc se nastavuje rádius přes metodu calculateRadius, která se nachází v package Calculation ve třídě CalculateRadius
3. Třída Planet udělaná stejně jako třída Comet.
4. Třída rocket dědí od SpaceObject a navíc nic nemá, nemá žádný rádius, tuto třídu nepoužívám, ale v budoucnu chtěl bych ji používat

## Package Calculation.

1. Třída PlanetHit udělaná tak, že má v sobě jenom jednu metodu, kterou jsem pojmenoval isPlanetHit která přivezeme Elipse2D, předěla ji na Area abych mohl používat tam contains a také spocte nás translate a zatím pomocí translatu a scale změní místo našeho kliku a ověří, zda hitnuli jsem planetuudělán přes getry a setry a udělán konstruktor pro vytváření vesmírného objektu.
2. Třída MyData, použije cestu, kterou předávám přes parametr v args[0] udělá a předěla pole Stringu na list Stringu, zatim rozpracuje všechny data a udělá listPlanet, Comet a Rocket také, nastaví konstany, taky ve třídě se nachází většina dát, kterou používám pro funkčnost programu například jako worlLength a pod.
3. Třída CalculateRadius, spočte radius pro planetu anebo kometu podle formuly, **„Math.*pow*( ( (3\* mas ) / (4\*Math.*PI*) ), ( 1.0/3.0 ) );”.** Tuhle formulu našel jsem v internetu, hustotu planet nastavil jsem na 1, a také počítám se s tím, že váha planety může být záporná, aby to šlo zobrazit, fakt nevím jak v životě může stát, že vaha bude záporná ale měl jsem to v příkladu *negative*, takze počítám se s tím
4. Třída CalculatePlanets je tady nejdůležitější věc pro výpočet světu a vubec fyzického chování planet ve vesmíru, metoda UpdateSystem přivezme parametr t, co je v podstatě uběhly čas od poslední aktualizace systému, dal spočte krok s kterym bude prochazet tuto smycku, spočte to tak že vezme simulationTimeTick, který jako většina atributu se nachází ve třídě MyData, a vydeli to 10000, a tu 10000 vydeli ještě předtím poctem planet, protože pokud uděláme přibližně 10000 kroku za vteřinu pro 500 planet, tak počítač to neutahne, takze to je moje vlastni optimalizace tohoto kroku, zatim to ve smyčce while proběhne a spočte zrychlení pro všechny planety a udělá to tak, že vezme pozice planety anebo comety v ListuZrychlení přidá to do metody computePlanetAccelrarion - tato metoda udělá jenom to, že podle formuly



spočte zrychlení pro planetu, zatim to vrátí se do smyčky while v metodě UpdateSystem, aktualizuje ruchlos ale jenom polovinu zatím pozice a zatím tu druhou polovinu rychlosti, 

toto uděláno pro lepší chování planet

, no a kdy se ta smyčka ukončí tak se ten svět zaktualituje metodou calculateWorldSize() tato metoda udělá to, že najde nejmenší a největší souřadnici X a Y ale také bude se počítat s rsdiusem planety, zatim vezme nejmenší a největší souřadnice a podle nich spočte délku šířku a střed vesmíru tohle uděláno, proto, abych mohl zatím ve třídě DrawingPanel použít scale. Pridany metody checker, tato metoda dělá to, že ověří zda byl sraz mezi planetami anebo ne, pokud jo, tak spusti ještě jednu metodu, která se nazývá start Collision, ten start Collision udělá to, ze spočte inerci a zatim spočte nový radius, váhu, rychlosti planety a planetu s nejmenší vahou smaže, také pokud náša planeta byla vybraná spustí se metoda changeHitStatus, a planeta která byla výpravná strati svoji funkce ( resp. pokud graf byl otevřen, on se zavře). Také byla přidána metoda updateLastPositionInfo ktera prida do fronty planety informace o jeji pozicích za poslední sekundu

## Package Main.

1. Třída Galaxy\_SP2022 je hlavní třída tohoto projektu přes parametr příkazové řádku přidává se soubor.scv který má být spuštěn pro simulaci, zatim data se spracujeme pomocí metody produceData(), take v třídě je mouse listener, ktery poslouchá klik na planetu, a také keyboardFocusManager, který zjistí zda byl zmáčknuty mezerník nebo ne, na a ještě tam je časovač, který aktualizuje obrazovku 100 krát za vteřinu
2. Třída Drawing Panel je třída která je využita pro vykreslování objektů na obrazovku, tato třída počítá se zmáčknutym mezerníkem, zmáčknout planetou, vykresluje simulační čas, všechny planety co jsou na obrazovce, také vykresluje text pokud byl zmáčknuty mezerník, taky pokud na planetu někdo zmáčknul vykreslí info o planetě a obarvi ji červenou barvou. Byla přidána metoda drawPath, která kreslí trajektorie planety, to kreslení v podstatě jenom použití kreslení různých elips s různou barvou a velikosti.
3. Třída ChartPanel zajistí kreslení nového okinka pro graf, také zajistí prekresleni nového grafu a jeste zajistí obnovení informací která se dává do grafu.
4. Trida Informer, veškerá třída, která je udělána navíc, za tuto třídu nepotřebuji od Vás body, tato třída dělat to, že generuje různou informace o vesmíru, generování uděláno jak generování bez opalování.

## Nejtěžší problémy co jsem řešil.

1. Setkal jsem se s tím že Java velmi citlivá na závorky. Kvůli tomu, že měl jsem špatně udělaný závorky v fyzickém modelu, ten model fungoval špatně.
2. Ještě podle mne přišlo těžkým spočítat správně svět, abych mohl udělat ten scale, nepočítal jsem s radiusem planet, ale za hodinu přišel na to, že musím s tím se počítat.
3. Takže měl jsem problém s časovačem, který počítá simulační čas, při zmáčknutí mezerníka ten čas mne furt se počítal, opravil jsem to tím, že začal počítat čas, který uběhl v stavu (paused system).
4. Měl jsem problém se zmáčknutim na planetu, špatně počítal translete planet a kvůli tomu můj program nefungoval na příkladu collision.
5. Měl jsem problém s tím, že pokud udělám krok 10000 za vteřinu a dám tam scénář random500 tak můj počítač to netahl a měl jsem maximálně 10 fps, optimalizoval jsem to tak že začal jsem počítat tek krok podle počtu planet v mém Listů planet.
6. No a podle mne největší problém byl v napsání této dokumentaci, kvůli tomu, že nejsem Čech, nemám českou klávesnice na počítači, takže psal jsem to přes telefon a opravoval jsem to na počítači, takže se omlouvám za gramatické chyby...

## Závěr

Podle mne tato první část byla velmi zajímavá, pokud bych pracoval v teamu, tak ten code bych napsal ještě líp ( víc přehledně ). Doufám, že nemám v první částí chyby a tato semestralka nebude mně vrácena

*S Pozdravem*

*Andrei Akhramchuk*