# Styl „Nadpis1“ pro kapitoly

## Styl „Nadpis2“ pro podkapitoly, kdyby je někdo náhodou potřeboval

Styl „Normální“ pro běžný odstavcový text

Styl „Počítačový kód“ pro mimořádně zajímavé části kódu.

# Co psát do dokumentace?

Představte si sebe v situaci, kdy jste obeznámeni se zadáním semestrální práce a najednou se vám do ruky dostane projekt vašeho kamaráda v nějaké fázi rozpracovanosti. Vaším úkolem je projekt dokončit, tj. navázat na to, co je již k dispozici. Co by vám vaši úlohu usnadnilo?

Mít JavaDoc dokumentaci API je jistě užitečné, ale možná jste již zjistili, že se v JavaDoc dokumentaci snadno ztratíte, když nevíte, co přesně hledáte. Uveďme si příklad. Víte, že potřebujete upravit vykreslení grafů, ale v JavaDoc dokumentaci vidíte hned 15 tříd. Která z nich se asi tak stará o vykreslení grafů? Něco lze odhadnout z názvů tříd. Někdy vám nezbyde nic jiného než si projít popis hned několika tříd. A někdy se řešení ani nedoberete jednoduše proto, že třída, která se o vykreslení grafů stará, je privátní / interní, takže se v dokumentaci API vůbec neobjeví. Pak zbývá jen zkusit si tipnout, jak by se mohla jmenovat metoda / třída, která se o vykreslení grafu stará (asi by mohla mít v názvu „graph“), a prohledat zdrojové kódy na výskyt tohoto jména. Dost pracné, co říkáte? Nebylo by super mít nějakou další pomocnou dokumentaci, která by vám pomohla s prvotní orientací v projektu? A právě k tomu slouží tato dokumentace.

Dokumentace by měla obsahovat:

* stručný ale výstižný a přehledný popis implementovaného řešení (např. „hodnoty ABC za posledních 100 sekund se zobrazují spojnicovým grafem; graf se plynule překresluje, jak přicházejí nové hodnoty ABC“) včetně toho, jak je to na nejvyšší úrovni dekomponováno do tříd a balíčků (např. „řešení je postaveno na softwarovém vzoru MVVM, … Nejdůležitějšími třídami ViewModel jsou XYZ, která se stará o veškerou interakci uživatele pro výběr dat pro zobrazení v grafech, … “)
* popis zavedených opodstatněných omezení a zjednodušení (např. „je použito sekvenční vyhledávání prvku v poli, protože typická délka pole nepřesáhne 10 prvků; pokud by se v budoucnosti pracovalo s podstatně většími poli, lze zvážit použití binárního vyhledání“)
* popis klíčových algoritmů za účelem objasnění chování, přičemž tento popis, často zapsán v nějakém pseudokódu, je na vyšší úrovni abstrakce, než je pak vlastní kód v programovacím jazyce. Rozhodně nekopírujte kusy kódy
* popis vytvoření, instalace a spuštění aplikace, pokud toto není intuitivní, nebo apriori dáno
* popis ovládání aplikace, pokud toto není intuitivní, nebo apriori dáno
* stručný popis dosud neopravených nedostatků a popis možného rozšíření do budoucna

Nezapomeňte rovněž vyplnit údaje v záhlaví! Celkový počet hodin = počet hodin soustředěné práce, tj. nezapočtou se do toho přestávky na oběd, večeři, vyřízení telefonu apod. Ale započte se do toho např. i pročtení tohoto dokumentu 😊

# Semestrální práce – Vesmír, část 1: Animace

## Simulace pohybu vesmírných objektů

Úkolem bylo vytvořit simulaci pohybu vesmírných objektů. Základní třídy pro tuto simulaci jsou:

SpaceObj – třída, jejíž instance reprezentuje jeden vesmírný objekt.

Space – třída, jejíž instance reprezentuje vesmír, kde se vyskytují všechny vesmírné objekty.

DrawingPanel – třída, která zajišťuje vykreslení vesmíru a jeho vesmírných objektů.

CSVLoader – třída, která nám načte data z csv souboru a vytvoří z nich instanci vesmíru.

Coord2D – třída, která reprezentuje XY souřadnice. Využívá se návrhového vzoru přepravka.

Galaxy\_SP2022 – hlavní třída, která zajistí inicializaci programu, ošetření uživatelských vstupů a opakované překreslení instance třídy DrawingPanel

Před začátkem simulace se nejdříve načtou data pomocí třídy CSVLoader. Tato třída pomocí metody parseDataToSpace načte data z poskytnutých souborů, které jsou ve formátu CSV a z nich vytvoří instanci třídy Space, která bude obsahovat její gravitační konstantu, krok v čase a všechny vesmírné objekty popsané v načteném souboru.

Třída Space obsahuje simulační čas, který se dá získat metodou getSimulationTime. Jeden z funkčních požadavků je pozastavení simulace. Ve třídě Galaxy\_SP2022 je nastavený KeyboardFocusManager, který detekuje stisknutí mezerníku. Při stisknutí se vyvolá metoda startPause/stopPause, která nám zastaví/obnoví simulaci.

Pozastavení nám zajištují metody startPause, getCurrentTime, stopPause. Jelikož je simulační čas závislý na metodě System.currentTimeMillis, musíme začít počítat uběhnutou dobu od pozastavení simulace. Tuto dobu odečteme od simulačního času a takto „pozastavíme“ čas. Aby pozastavení fungovalo vícekrát, musíme odečítat od simulačního času sumu všech „uběhnutých dob od pozastavení simulace“. Například: první pauza trvala 3 sekundy, druhá pauza trvala 7 sekund, třetí pauza trvala 4 sekundy… Suma těchto dob je 14 sekund a toto odečítáme od simulačního času.

Simulační čas je také ovlivněn krokem v času, který určuje kolik simulačních sekund proběhne za sekundu. Metoda getSimulationTime() toto zajišťuje vynásobením simulačního času v sekundách tímto krokem v času.

V této třídě je také implementovaná simulace pohybů vesmírných objektů. Metoda updateSystem si načte uběhnutý čas od poslední doby, kdy byla tato metoda vyvolána a provede kalkulaci všech pozic a rychlostí za tento čas. Předtím než se vypočítá budoucí pozice a rychlosti, tak se vypočítá zrychlení všech vesmírných objektů pomocí metody getAcceleration.

Třída DrawingPanel si pomocí konstruktoru uloží referenci na instanci třídy Space. Zde zajišťuje vykreslování planet metoda drawPlanets. Úplnou viditelnost objektů v každém čase a maximální vyplnění dostupného prostoru okna nám zajišťují metody getScale a getMinMaxBounds (které se ale již vyvolává v metodě getScale). Metoda getMinMaxBounds nám vypočítá levý horní roh a pravý dolní roh, které ohraničují náš vesmír. Obdélník stvořený z těchto dvou bodů reprezentuje nejmenší možný obdélník, který v sobě obsahuje všechny vesmírné objekty. Tato metoda využívá i velikosti vesmírných objektů pro vypočítání těchto bodů.

Metoda getScale nám vypočítá vhodnou hodnotu pro metodu Graphics.Scale se kterou můžeme vhodně vyplnit dostupný prostor okna se zachováním poměru stran. K vycentrování vesmíru využijeme metodu Graphics.translate a atributy offsetX a offsetY jako parametry této metody, které jsme vhodně vypočítali v metodě paint. Pomocí getScale a attributů offsetX a offsetY jsme si také zajistili responzivitu našeho okna.

V pravém horním rohu okna se vypisuje aktuální simulační čas pomocí metody drawTime, která využívá metodu getSimulationTime z instance třídy Space.

Poloměr objektů určuje metoda getRadius ze třídy SpaceObj. Předpokládá se jednotková hustota všech objektů a využívá se vzorce pro objem koule.

Ve třídě GalaxySP\_2022 byl také přidán MouseListener na instanci třídy DrawingPanel. Při stisknutí levého tlačítka na myši se vyvolá metoda getSelected(Coord2D coord), kde parametry jsou relativní souřadnice (vůči panelu) myši při stisknutí reprezentovány instancí třídy Coord2D. Metoda getSelected detekuje, zda tyto souřadnice nejsou obsažené v jednom z našich vesmírných objektů. Pokud ano, do atributu DrawingPanel.selected se uloží reference na instanci této třídy. Metodou drawSelected se pak vypíšou informace o tomto objektu v prostřední dolní časti okna.