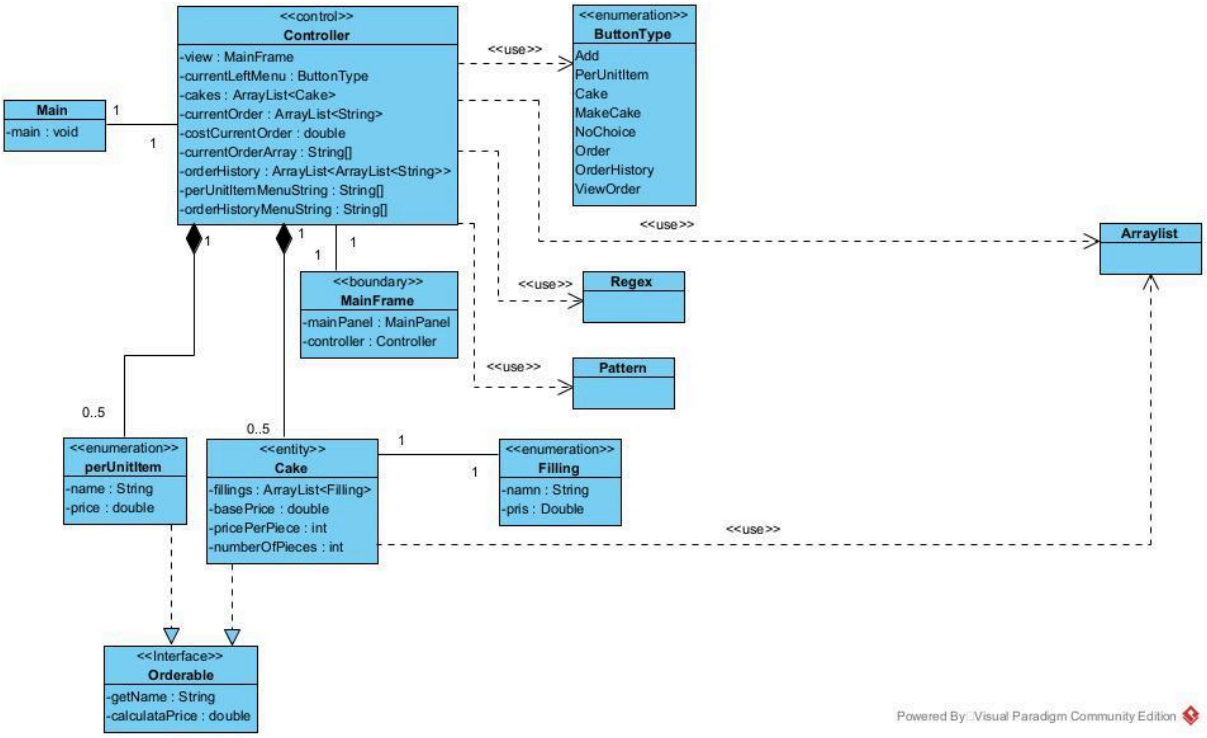
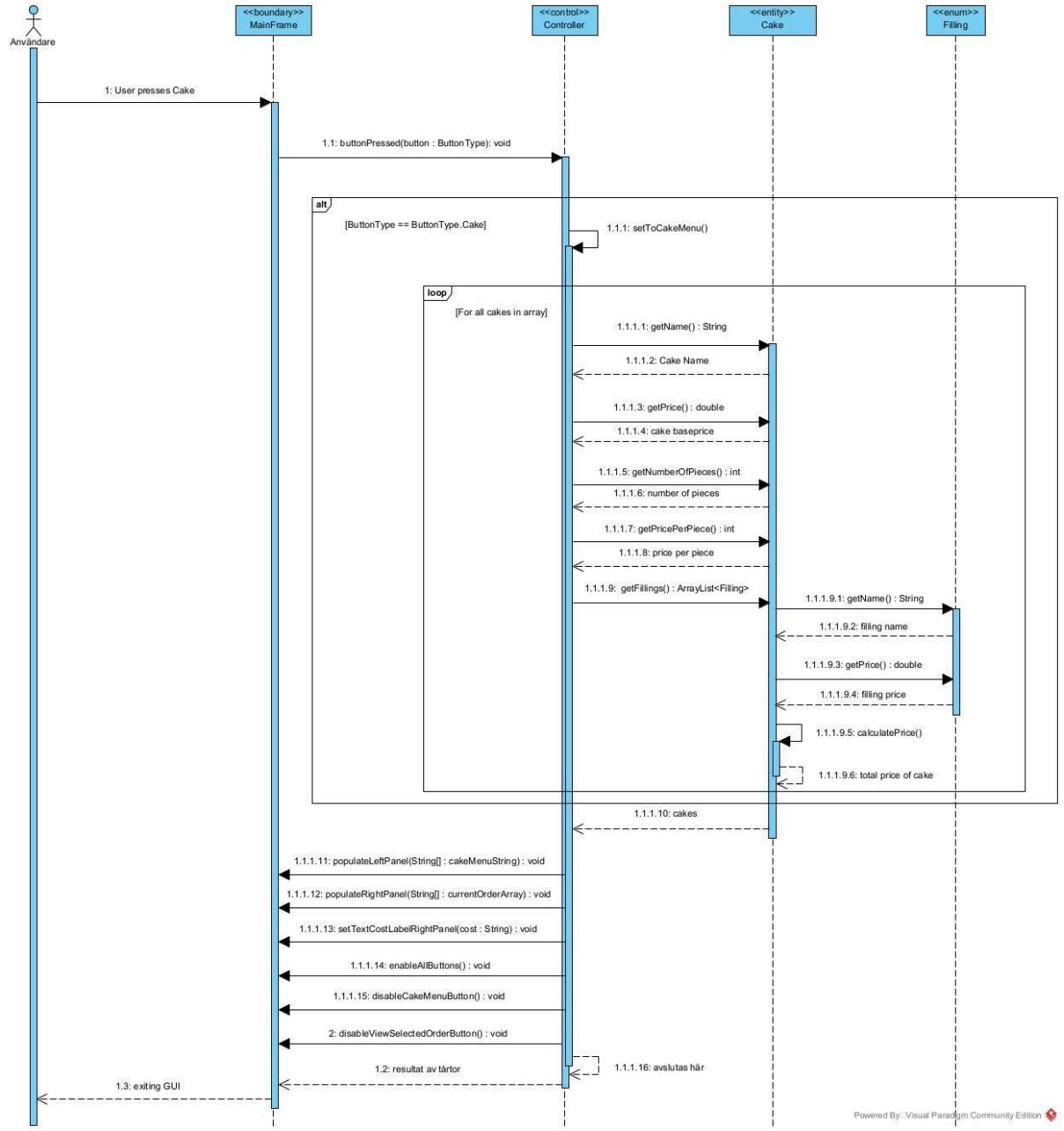
Alper Eken AQ0647
Josef Zakaria AQ1637
Datum för inlämning: 2024-12-15.

DA339A Rapport U3 Grupp 4

Klassdiagram:



Sekvensdiagram:



Reflektioner

Reflektion över egen design

Vi började med att skapa en interface som vi kallar för Orderable, där vi hade metoder, getName och calculatePrice. Anledningen till att vi skapade den var för att dessa metoder ska finnas med i Cake- och UnitPerItem klasserna. På grund av att vi hade denna interface var koden mer finslipad och vi hade mer ordning och reda. Fördelen med en interface är alltså att koden blir mer flexibel, även om vi endast har med två metoder. Detta är på grund av att om vi skulle lägga till något mer som exempelvis en klass för dricka, skulle vi endast kunna skriva “implements Orderable”, så skulle det gå hur bra som helst.

När vi designade vår program bestämde vi oss dessutom för att vi skulle använda oss utav enum för exempelvis filling och unitperitem. Vi lade till dem som enum så att de olika fyllningarna och unitperitem skulle finnas med i koden, vi gjorde dessutom så att vi lade till priser för de olika enumarna. En nackdel med att vi använder enum är att om priset skulle behövas uppdateras, då måste man gå in i koden och ändra priset manuellt, vilket kan leda till att något råkar bli fel. Vi funderade på att använda oss av en abstrakt klass också, då cake och unitperitem skulle kunna ärva saker som namn och pris, men vi bestämde senare att vi tyckte att en interface passade mer då vi kunde ha metoder istället för att endast attributer. Vi valde dessutom att använda oss av ett interface för att programmet inte skulle få problem om vi skulle lägga till saker, om vi skulle lägga till fler saker i en abstrakt klass så skulle det innebära att vi behöver redigera våra konstruktors i våra klasser som ärver den abstrakta klassen, vilket skulle vara för cake och unitperitem.

Reflektion över skillnader mellan egen och andras design

När vi hade vår workshop för U3 diskuterade vi vår klass- och sekvensdiagram med två andra klasskamrater och de var i en grupp. När vi kom till workshopen var vår sekvensdiagram väldigt lika och detta var då på grund av att vi inte hade kommit så långt i vår kod och de personerna som vi var i grupp med, hade inte börjat med deras kod ännu, de sade att de endast hade gått igenom kodskelettet. Men även om vi inte hade kommit långt under den tiden, så hade vi dock ett par skillnader. I sekvensdiagrammet använde vi oss av väldigt många getters och setters, men de hade använt sig av en toString metod, som skickade allt det som behövde såsom namn, pris osv. Klassdiagrammet såg däremot väldigt olika ut, vi hade ett väldigt simpelt klassdiagram som var enkla att förstå, då vi hade gått igenom kraven och sett att vi inte behövde alla klasser i View och att vi inte behövde ha med metoderna. Vi rekommenderade dem att ta bort alla klasser som var onödiga och att de inte behövde ha med attributen. Men annars var de väldigt lika då både vi och dem inte hade kommit så långt i vår program.

Frågor till andra grupper

Fråga 1:
Hur valde ni att strukturera era fyllningar och styckprodukter, använde ni enum eller klass eller annat? Varför valde ni detta?

Fråga 2:
Vilka felhantering har ni i koden och varför tycker ni att det behövs just där?