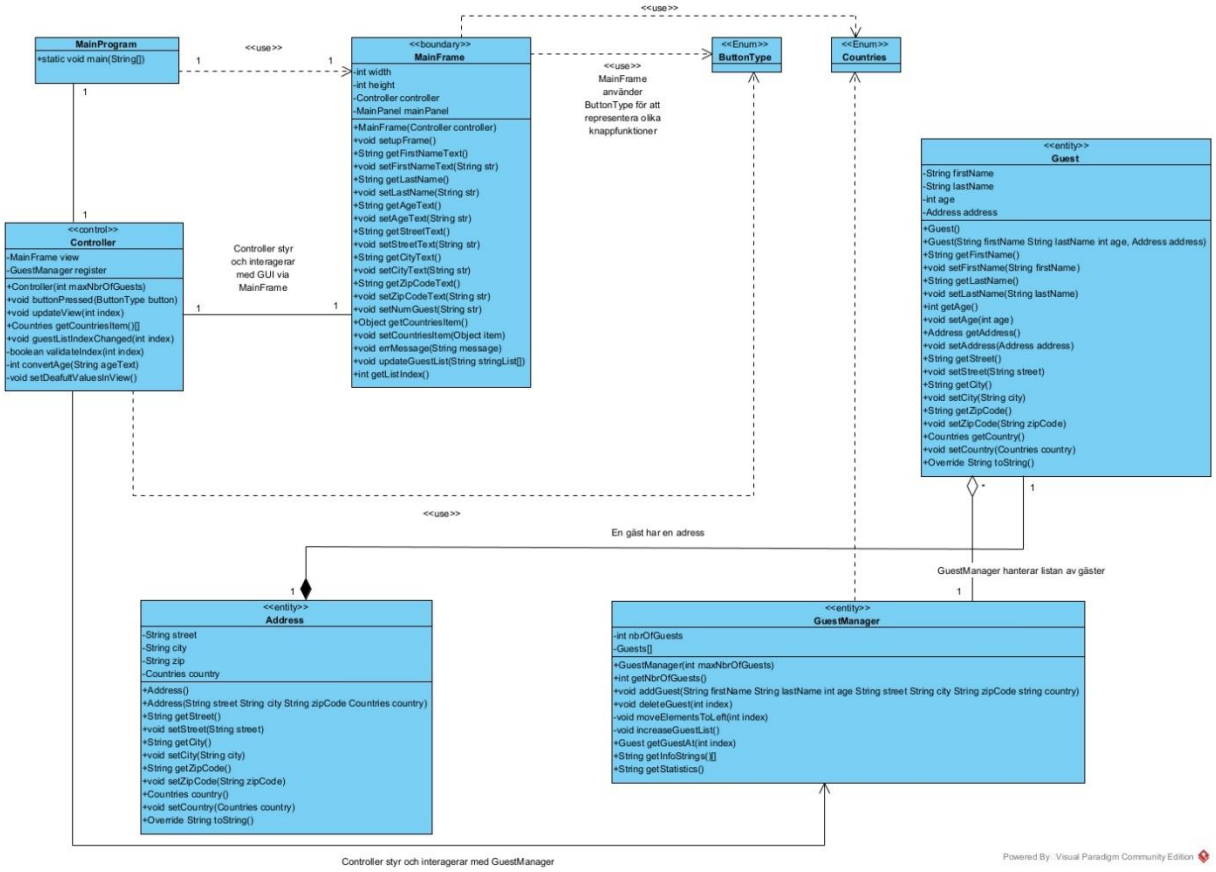
DA339A Rapport U2

Namn: Josef Zakaria

Datorid: aq1673

Datum för inlämning: 2024-11-17

Klassdiagram



Reflektioner

Reflektion över uppgiftens utmaningar

En sak som jag kände var väldigt utmanande med hela uppgiften var att få alla delar att fungera ihop. Det var många gånger jag behövde gå tillbaka och fixa med koden, speciellt i Controller, som får allt att arbeta som en helhet. En annan jobbig utmaning var att få valideringen av användarinmatningar att fungera, jag fick verkligen lägga ner tid på att förstå vad som var fel och fixa det.

En sak som också tog tid var att förstå hur allt skulle kopplas ihop. När jag väl fick i gång interaktionen mellan Guest, Address och GuestManager, kändes det som att hela projektet började falla på plats. Detta lärde mig att hur viktigt det är att testa koden steg för steg.

Reflektion över uppgiftens implementation

Att implementera arrayen i GuestManager och få metoden moveElementsToLeft att fungera var något jag kände var svårt. Det är lätt att glömma och göra små misstag, som att justera arrayens storlek eller flytta elementen på rätt sätt.

En annan utmaning var att få valideringen av inmatningar i Controller att fungera. Det tog tid att skapa en logik som täckte alla situationer, men när jag väl löste det kändes det riktigt bra. Jag lärde mig också hur viktigt det är att ha tydliga felmeddelanden och att kunna återställa GUI när något går fel.

Även om jag kände att det var svårt så tyckte jag också att det var också roligt att skapa relationen mellan Guest och Address. Det organiserade programmet och gjorde allt mycket mer flexibelt.

Reflektion över skillnader mellan U1 och U2 för hur gästlistan hanteras

I U1 jobbade jag med en enkel array av strängar för att representera gäster. Det kändes som något väldigt grundläggande system, men när jag började med U2 och använde objekt för att hantera gästerna såg jag tydligt skillnaden.

Att använda sig av objekt gjorde allt mycket mer lätt att arbeta med på grund av strukturen. Varje gäst hade sin egen klass med egenskaper som till exempel namn, ålder och adress. Det kändes som att koden blev mer logisk och jag tyckte också att det såg bättre ut visuellt. Att använda metoder i klasserna direkt, som i Guest och Address, sparade tid då jag slapp upprepa kod. Och att lägga till nya funktioner som statistik i GuestManage*r*, gick också smidigare på grund av att allt var strukturerat väl.

Att gå från en array till objekt gjorde programmet mer komplext, och det var mycket man skulle hålla reda på. Det tog tid att förstå hur klasserna och deras metoder skulle arbeta tillsammans. Jag märkte att när flera klasser samspelade ökade också risken för buggar. Ett exempel är interaktionen mellan GuestManager och Guest, där det var viktigt att varje gäst hade rätt adress kopplad. Även små misstag kunde orsaka större problem, vilket gjorde att jag behövde lägga ner mycket tid på felsökning för att få allt att fungera på rätt sätt.

Men allt extra arbetet gjorde så att resultatet i U2 kom fram mycket bättre. Koden blev mer flexibel, lättförståelig och kändes enkel att underhålla. Det kändes enkelt att lägga till nya funktioner utan att ändra på hela programmet. Jag känner att U2 har gett mig en djupare förståelse för objektorienterad programmering och vilka styrkor den har. Uppgiften gav mig också insikt i hur viktigt det är att planera koden från början. Med en stabil grund blev det mycket enklare att bygga på koden och göra ändringar utan att riskera att hela programmet krashar.