Analys miniräknare:

I programmet calculatorAssignment har jag använt mig av först en while-loop där jag satte variabeln bool isRunning till true och gör så att while loopen körs medans isRunning är true eller sann. Inom While-loopen körs metoden RunCalculator som i sin helhet är hela programmeringen för miniräknaren, det körs även efter att RunCalculator har körts en fråga där man frågar användaren om den vill kolla sina gamla uträkningar, skriver spelaren j eller J för inom if-satsen har jag gjort så att när man skriver ett litet j blir det ändå ett stort J genom Answer.ToUpper() == ”J”. Om man skriver j skriver den ut listan som har skapats som samlar in de uträkningar som användaren gör, det gör den med hjälp av en foreach-loop. Listan skapade jag som en List<T> utanför den första metoden, detta gjorde jag så att den skulle kunna användas i båda metoderna och även för att ha koden mer strukturerad. Jag valde att använda mig av List<T> då den är lättare att lägga till och ta bort objekt när listan från början är tom till skillnad från en Array[ ]. Sedan skrivs det ut ”Vill du avsluta programmet?” efter det att användaren har svarat på den frågan och om användaren skriver J eller j sätter den isRunning till false vilket avslutar loopen och alltså programmet.

Inom den andra metoden som finns i koden skapade jag först tre doubles vid namn num1, num2 och result, jag skapade även en char vid namn operation som används längre ner i koden. Det första som skrivs ut med hjälp av Console.WriteLine(); är en välkomnande fras. Därefter skrivs det ut ”Skriv in det första talet”, jag valde att använda mig av do{} while(), detta gjorde jag genom att den skriver ut frasen om och om igen medans det som användaren skriver in, vilket är variabeln som jag skapade tidigare: num1 inte är en double. Jag valde att använda double då man ska kunna skriva in decimaltal. Efter det skriver programmet ut ”Skriv in det andra talet” och därefter gör den samma sak igen men med variabeln num2. Till sist skriver programmet ut ”Välj en av operatorerna: (+, -, %, /, \*):” och då tar programmet in en variabeln som hette operation som var en char, den tar in en av operatorerna som står inom parentestecknet.

Därefter använder jag mig av en switch-case som kollar igenom operation, alltså den operatorn som användaren skrev in. Jag skapade 5 cases där svaren är de olika operatorerna, inom operatorerna räknar programmet först ut svaret av num1 och num2, alltså de två talen som användaren har skrivit in. Svaret sparas i variabeln som jag tidigare skapade vid namn result, därefter skrivs result ut i konsolen och det sparas även i listan som jag nämnde tidigare vid namn mathCalcList. Den läggs till med hjälp av funktionen mathCalcList.Add(saveOperationCalculation). Det som står inom parentesen är stringen som jag skapade för varje operators case och i de stringsen sparade jag användarens kalkylation. Namnet på stringsen beror på vilken operator som användaren skriver in, till exempel om det är ”+” heter stringen savePlusCalculation. Jag gjorde även en default så att om användaren inte skriver in en av operatorerna som finns inom parentes tecknet skriver den ut en röd error text som säger till användaren att skriva in en giltlig operator, efter det startas metoden RunCalculator() om.

Gränssnitt:

Gränssnittet inom programmet är användarvänligt på det viset att det tydligt står vad användaren ska skriva in, när användaren ska skriva in vad osv. Till exempel när användaren ska skriva in det ena talet som den vill skriva ut står det ”Skriv in det första talet” och när det är det andra talet skriver programmet ut ”Skriv in det andra talet” detta är väldigt tydligt för användaren att förstå vad de ska göra och vad de ska skriva. Jag har även gjort så att om man inte skriver in ett tal utan en bokstav eller någonting annat kommer inte programmet att gå vidare. Detta gör det tydligt för användaren att den måste skriva in någon siffra eller decimaltal för att ens kunna komma vidare i programmet. Det är även väldigt tydligt när till exempel programmet skriver ut ”Vill du avsluta spelet?(J klicka annars vilken tangent som helst)” detta gör det väldigt tydligt att användaren ska skriva J eller kan också skriva j om den vill avsluta spelet, samt. Att man annars kan klicka på vilken tangent på tangentbordet som helst. En annan sak som är tydlig är om användaren inte skriver in en av operatorerna som den kan välja mellan kommer programmet att skriva ut ett rött error meddelande som säger ”Error ogiltlig operator! Försök igen.” Detta blir extra tydligt, speciellt med den röda färgen, det blir också tydligt med hjälp av texten som skriver vad som blev fel när användaren skrev in.

Mönster och strukturer i koden:

I koden återanvänds while-loopen som loopas medans isRunning är true, switch case används även och går igenom variabeln operation, den skriver ut svaret och sparar svaret i listan. List<T> används även genom hela koden då den sparar de olika uträkningarna som användaren gör, den skriver även ut uträkningarna som användaren har gjort.

Förbättringar:

Om jag hade fått en möjlighet att göra om uppgiften eller utveckla den mer hade jag börjat med att lägga till så att användaren kan skriva oändligt med tal och addera eller subtrahera, precis som man gör på en riktig miniräknare. För att få en liten utmaning och även för att det ska bli mer användarvänligt hade jag försökt att skapa ett grafiskt interface som har en miniräknare i sig. Jag hade även gjort så att om användaren skriver in någonting annat än en siffra så hade ett error meddelande kommit upp för att förtydliga vad det är som blir fel när den inte skriver in en siffra.