**Dokumentation: Loggbok**

**Planering**

Jag har använt både [pseudokod](https://docs.google.com/document/d/1pV0_AzzFoQWL7Y5ZCQN5ButCDnmxUq0nTEdcnVWam6s/edit?usp=sharing) och [aktivitetsdiagram](https://app.diagrams.net/?tags=%7B%7D&lightbox=1&highlight=0000ff&edit=_blank&layers=1&nav=1#R%3Cmxfile%3E%3Cdiagram%20id%3D%22C5RBs43oDa-KdzZeNtuy%22%20name%3D%22Page-1%22%3E7R1bm6JG9tf42P1RQHF5nJ5JspOd7Ga3s5uZRxS0SSN0EHva%2FPpUCSicUwoit5bqlxbEY1nnfq2Z9nH99lPsvDz9ErleMFMV922mfZqp7M8g7B%2B%2Fs0vvmJqV3ljFvpveIscbj%2F5fXnZTye5ufdfblB5MoihI%2FJfyzUUUht4iKd1z4jj6Xn5sGQXlb31xVh668bhwAnz3d99NntK7lmoe7%2F%2FD81dP%2BTcTw07fWTv5w9kv2Tw5bvS9cEv7YaZ9jKMoSV%2Bt3z56Ad%2B8fF9%2B%2F7z7PfjybPz08382fzr%2Fe%2Fjnb%2F%2F6%2F10K7MdLPnL4CbEXJu2CVlPQr06wzfYr%2B63JLt%2FAONqGrseBKDPt4SlZB%2BwlYS%2F%2F8JJklyHc2SYRuxXFyVO0ikIn%2BBJFL9lzyyhMsscIv%2FZC9wNHLLueB9HiOb31ox8E2Xewq%2Bx5i11tkjh6PuCOAzgggj8cOHMveHAWz6v9Qj9GQRSzt8Io9DgolxFD9luOi%2FvhePeh5t5mOODwCnSV7fRPXrT2knjHHoi9wEn81zLxORkNrw7PHT76a%2BSz71WVjN80ot8b6YcyfjMU856WwWyibbzwsk8WMV4NjCJgiROvvAQBYy8Kv%2Bt4a09UFxAYrSawahy9axJsmcB2ZfHalN7yR6LlcuOdJ6M7LZeA2XdTat%2BrzUjyTjUrYbVHkf8Ovn396%2BsPj799sX8myuev6vMfdiYlixT5yL4xcWaqETAEPbj%2BK3u54i9f4mgVO%2Bs136D0TfaNhfcxKQcB02sc69%2Bf%2FMR7fHEW%2FJ3vTLWWCdfZvKTKbum%2FccJ%2BWDLaywlnpmou9SxXP5Be4R1LnWuGcTFdvXpx4r2dpawcRboOUKRk19%2BP%2BpPkJPFU1J35TRH9ldB4Kc4IwtmH102wHRnSltbCWyxESJtbVKdKl0gjgEc1OjjSNMxozzHHg7LlEmHthdsKY6MCH3D7l0tVvP2uMTdopzyDxJom2H4i2H6jq93HivfL7KM2%2B6Bv%2BCfDFAE79jbCAdNtL%2Fwl2yYnCLxgz06csbzYZ2vzYvjer8c3qlHGOCfXxBiFHmGCzxSh0DZMzekUhZCDhCgUcVBnKDQEtlMq0bg9s%2FeZ8v0x%2FtxGqXBjbDCnnBqPtzK5Rx%2FMGf1UEIoplFNSUVplF1hlRRIr2GhXGmkZVEstgdWBvVTHhCMGNruUpjbcQZAVgNF%2BjThTwBrT8lsZ3ca7rxwY8%2Biyy2%2FZF%2B0vPr0dlsWvdtlVhuVcZHNR7Yer%2FE1uCWe%2FgQtb781PvuZg2Otvx%2B9jV8dv4Bf5F1zGWelyzmA6izal9FRtbdTlVOVeY38lMlb7c6lUWuYgwwY6pC4vqsBWNywAqGM%2BtM7beMxGYCa6wthltbc6gtQEWa1uxfBjsq%2B8%2F8bQRgPBvtKVVoN6gdXAti0BTlPgr0L2esF2fm8h8s31F07wIXtj7btuak94G%2F8vZ74HxYnghRPtfnvoA18Cg8Wk9eZoPJbRnUnOIm000vX1ka9pgPlUjHxdgHutM9xjl2tqdt1RLRaVYkFHnlCLl9FIpTbKKaFSHRVIhQpIhbZjQWqWXrbYoPipq28OsaBTgE7oG4Z%2FZ1d4LOPtDjQS0WvwwJEgM7IpUHmZ%2FC%2Bji5GgG2LJglZBU3QfxFlP5gWR6YOWCVDoqBpWO1KmDNUuf76GWYzITbOb5r4oHZhyRcEbSbmtU67ZDuUCeXk15TaXuBQBqpmxPa1i6y6ZnF%2BZSs89z16kK2iXjbCDKdloOmxkwuRUUzZigHpiI1O9jI3A8x2xkS3ZaMJspCstsRED1BMb6cplbASe74aNcmkkCHeGu4SvrhTiVPxZGvtceSFit3ca8IQUMXyiO%2BevIlI4Dh6M5ykixBIVfvSLEBUjhKkDL77tVADyOMngiMDh4P96rr%2BaGiY0c3BM4KDkb3scML6YFCoOic7hUCEqHbgqP6Zdnh8TZq5QpkyQyiog%2BWQeTUQc1wS162Nbt6sZzxAgu7OEmCoKH1yFbF0i%2BwSyRazdL7JFTu5VyKYS2SeQLbIye0V2Drg9ZBsS2SeQLbJk%2B0U29ihgMXPCNkuWMqfvmjBJP3gls1YnKS%2Fjj1fHH7OkY7t53MvjjwasNm5abmLCSmOjbvyxrZCfJisQeqHckUTOLUxwzSjXwizQN%2BXiCgSkNLkZI3VmyaAdj87EgYpHvu9ZvKjVgFFXu0oUyE%2BDpws0WVEwJYFut2WKYFLuX6LXyOK%2Fa8LsfVrHKJqBiAX8NaI0zdffqXn586nGoo5JVBd1n0jpOhVHzzJB0W9TR48B6lm45l21knInSbmoPrwp5TJAHZVI4SWfL5EyLHru%2BW5KpPT%2Bp34sFrYt8hk1Q7M1t8s4KxRag5fe6DjOOpndH77eRpexwimpEEttSYVYZl8qBC%2F5vAqx9CFUiGydmjIbNfYhIBvV9yGuZaO826suG4HnO2IjHD%2BWE8DOsQdM2QjNuV5TADqOVUsUXoJCoU3YLwrrdF7d%2BDgCosAwflMRb2MRf6WAbzuMlG9ZTXznlsStIdyGk6dVRbnXmyFdAI0Yyr1S%2FNPGRgZ1JvFMje3tpt28WID0PM2Byvq1lklw1C5Jm5RrIUgdOSWCRZ937okChteBD3TjllAZJJs0KyHR3ZiVrM7CZIJFX8hKVh%2BBMioDZZNiJYLjxA1ZScWh665YCS%2B6gpWIPUDMmdYYKC1Z6WZYCcYRVM26t6l9%2FAMg6%2BsoyFiafa%2BfAdwSl%2BHfY4t%2Bz8ntgCx39uMdcaBglPRxjIe35JFJVfn8CbHlO01fY3Nm8NZ4imOVAhSIO%2B1uBguDVxEYOIJ4HJ%2BSI8F1ku36ZpAAT89RBy%2BkyU%2B9PIuE2%2BIEiAQy%2BNQUQ5VGWR9G2UhaNZjlD0A0H%2F9L4MEYfUeMDTmKf9K0a5P2aBdFn7umXZntmBTtomhQ0%2FYjbMjVnhfaGu3K9MKkaFez2qJdXRuadk1sMxTKvTaZKx7y49ogSU%2Bz3Iuo0HsfvGTPlLpzUvInz05cffADYSYjhNS3%2FJG6c1K0q0F%2Fu2lZJoFHxvffvmvWyYXfesFdpeP4bsInJs7HRusZHOe%2B9MNwXwk%2FOzPW%2FSlaz7ebajMnjhK2%2BVGYUQCwclzHs5bCGcrGwvLmyy6tHMRdmppzV9HOEVGWTU8T0XV2TsXJr9HiiW9G7G32YzCdPXVvtgHbY4yiUYwqMuAuD56EMHE%2BTurjG9bHtDV9jGi5d32cm8GSdt%2BZDWHY%2Bj0oCyZNrQgD1JXgwUZdU2GdcUS3bhUSEN6ltHlqAo6qqnv%2Bb2sIVSVCW5wXNjw%2BRWnS6jHuimLbfOVwjPujwz8jZ7hzYwK2SqrYZeh1iHt%2BBPm0OyMpQepVNRuLYwytfxWLg4Y%2FOsHmeXL8BsYR2Tp2H%2FvlNxkRE6S2m4fnq5LkXfOZ7FBomQTHHQxokXJhUUbviSVLztyeEu3ewUnZ1Lq3jWbUq2PfVQeguiZeHIXdpyIOk3eUNCMRs1fPURx7z4II97tPQkA8kOFTEHm0VI5Db0uqXCk47gzlXi%2BNZSlPbaHQ1q091rwCsG72a4vZorCijFq0IGWg5lAHdqLsOiUxtx600Eg5xlAQ%2FpdyMgSl9Vzabk%2FdEn2P9KcqIEqNqOaC8zEAKUPJMfjkMPuEuelyo%2FtQ8bLhxbwuQ6fhrLl4Duebl1Sj7LWIywsxUg1SrpgpVcocNE7h8Rs0XcdouxKlTmbylgXRtFxi2B52jUs8Ap%2BYKKqAfKUF3EIewRqZBUyUy0ZZ3qYJTNozgSGovk1gouBEbJaxU%2B6gQbH2XJftHF9d%2BJraEaHLUcfu7M2SpRd4ON6V8Shix3HzHpCruoD1RCTWIeth73OCoUc0VInYYzDgaqRX37V5Vm19MjzHu68cGENHdvntsBB28emtdLVrxL%2BpODyDivzQo67Dm8SAdVeNjzU3yiqg71wyUURhEGm%2BtSCrDlIIonYw%2By0PqUpPcxqeJuqIx7X%2FzXviYRFC14KKyALuGli4ZsZBz%2FgUxQ2kMLrVOauY4BqPLNahWOtuZDFeNKlYG5j%2B3cvIYkKwH%2B%2BHyyhep%2B2W2GF3Er7ozX6OoR%2BmKYCq9EGwWzw7rsdzDmlL59M2nh17OfdJiWeHQ9%2Fwh%2BqlFEbR74kISjD5U%2B213%2FMQGZq0tqO2Dc8OIs3PDqr02QZP%2BBEy9Zz0tHQiPMZbs3kqqFDb1JDU4XnIms3TQqfhdnQqOf9awc85uRvgkPLzH%2B9Kk%2BKc%2B4N3RoU6THmmQFVlmY7l4kqxkGoXxMRHofUQkQw%2B5PfQnSrF3yTEHzw7ty3xB0%2F%2F7Un84Z9zkfiDJwMPIv5UHGSBx8wKzl2Y5shBeMKsUID2OnGQ5PJSCtBJCFB41gsXGcWzWmyzmQBFaVZdUe6VM4A7OruGf63g99Q9u%2Bb8x7uSoLhGJh1Xxi3ESeboNTqCHL1ap2f25kuXFMAhDDVNJQSEBFfTdfpAlRPRpqTq0OGBOp7J3biBX6%2BZ%2B7r6xMPjomufeYg%2B0pHe0rDvLWg28PZBj1IjAcoTCAoHD3CccOW%2Fhk6mDyfSdYDobRQaUcOunhSgNyxANewsNBSfuiDQ0VH%2BFS46l9RnBlkj8dmH8Kzjd9%2B8dali67K5ikb2Zc%2FlKZooASvrItuoiwRxM12Qbe%2B3LlITFXBLXLeAa218uMYp9o5aXm4UpzYoa9ZUoTGr9YtVHA2QWL1KKo8CrTp2SmV%2FGn9XBUw4fIOajt3Js4UT%2Bc19geI7KJfAXs%2Fw9RJ6HbdDuvA348JTDTneZSD1pwVbCFJXLjxcdKULT%2BkQLrx%2B9sRFP1wzsSXPWzw9VHwE1Q9596cUh9MQh4YlliyXi0MTy6iuxCFedEVHialAcdhHR4mOw1%2FZqWtHQTiNcgZIG6NI3uhyBPikRJ3VmqizT5hjrQs6wZKrct82EnX95L71Om03N5%2B%2BsRSYvtEa0hiC1HfyRhcVB8mAfhszyWBCdvCAfk7%2BMvTb%2BLhT6MqNIfRLcTxRovWiPA0I6BNjDFjFUZY6jeqx5%2Fqrfaf6qT71zUvaeX7Abi7D5wf78S6T%2BR%2F2boyf%2BE6QAgDS%2FtjnfpD4KfRR97BDu04YjzF7jcfQOvGYG%2B9hxwOr4QEFtWe9Vo6%2B7tqqoqKSGOl03qrTCdu5TdIw2wA7jRmgjqJraMnq%2BVwDbDkHz3fkblJcbjSRHnPDFiilfnOmVEbOpiTEYFN2YyEGHd%2FuhBhacoUQg43jPQkxHDOTneInWAN2igvFYL%2B5Uirb56YkBmFnNZdeRlvd4Uzg3JN%2BOsL5VxmXdYHjj3QkEXNmlZ3fx0G4I0iVGrLPbUqSjhDDRqLOaprJgp2bHJgGgHXVLFxYeO1uYfyZroTdiTEXsl24%2FXbhcYhRWVw3KTGqYTHaUIbCwvnuPGfULlzlOqN24X58Z0P2lnYkOmEX06F4b7DyBEP2lnaEa9hbOgJc49hyjex22fz5XI7qK6fNIeXJ4f5k4qx8LhSVOVOY2BwaQxoAqQBhAKzf5LQhR21Pyp6hdhlEc3vGUnqzZ6h9oT1jkEHsmTrB5Juvn4Uj8QyzadQBQYJHpXRd6WHiUOYEEara%2BiFIjl3xBiNtMLjaJ9e0h9o64dALUFu2Ll1v6WyD5N2ivJLzGmuJIVCtIlTLEtrLEho2ROLgFbQmDrxKpF7mNyLOHB6pouBqdZBguZzv27BhkGBGH8iMfpJxAv6uDniYCkaltBQmYJdxxDFxFOPslz79EjEuZDf%2FBg%3D%3D%3C%2Fdiagram%3E%3C%2Fmxfile%3E) som planeringsverktyg. Planeringsverktygen hjälpte mig att dela upp programmet i delar och därmed förstå vilka uppgifter jag har framför mig. De hjälpte mig också att hitta lösningar på mina uppgifter, eftersom mindre uppgifter alltid är lättare att lösa än en stor.

Redan i planeringsstadiet insåg jag till exempel vilka strukturer som är bäst att använda när jag skriver kod. När jag planerade programmet bestämde jag vilka metoder jag vill skapa. Jag har också lagt märke till vilka funktioner som är liknande varandra, så att om jag skriver en metod för en funktion kan en annan funktion göras snabbare med samma mönster (till exempel ändra och ta bort ett blogginlägg).

Jag tror att jag skrev programkoden enligt min planering, men min planering var baserad på min kunskap, vilket innebär att jag omedelbart var begränsad. Jag tror att när jag har mer kunskap kommer jag att kunna planera bättre. Det finns något i planeringen som inte fungerade som det skulle: jag har inte lyckats göra programmet lätt att använda. Jag måste tänka mer noga på meddelandetexterna till användaren och på hur jag kan förenkla programmet.

# **Utförande**

När jag började bygga programmet har jag skapat en lista [minBlog] som innehåller en vektor med strängar. Jag skapade en vektor [minInlägg] av strängar med 4 element: titel, innehåll (text), id, datum. Jag skapade en switch-struktur för menyn och en while-loop för att köra menyn. Jag skrev direkt hur användaren kan avsluta programmet, eftersom detta är relaterat till while- loopavslut. Jag placerade listan och vektorn på rätt ställen för att undvika fel på grund av variablernas synlighet. Jag placerade listan [minBlog] före while- loopen och vektorn [minInlägg] inuti while- loopen, men inte inuti switch-struktur.

Arbetet med programmet gick bra, men under arbetets gång uppstod nya problem, bland annat några kompileringsfel. Eller såg jag att jag ville ändra någon del av programmet. Till exempel bestämde jag mig för att göra de inre menyerna för att välja sorterings- och sökalternativ.

Det svåraste var att göra programmet användarvänligt. Det andra knivig sak är att hitta sina egna fel i redan skriven kod eller i programmets beteende. Jag har t.ex. felaktigt skrivit hur programmet ska välja index när användaren vill ta bort eller ändra inlägg. Jag skrev att index = id-1. Detta gäller fram till dess att användaren sorterar, och det tog jag inte hänsyn till. I detta fall ändras inte id, men index kan ändras eftersom inlägget kan flyttas till en annan plats efter sorteringen. Jag stötte på problem med de kyrilliska bokstäverna. Mitt modersmål är ryska och jag har ett kyrilliskt tangentbord installerat på min dator. När jag försökte skriva titeln och texten till inlägget på ryska, när jag ville visa inlägget och utföra andra åtgärder med inlägget uppstod fel och undantag.

Det finns något i programmet som inte fungerar som det ska. Till exempel är det reguljära uttrycket för sökning efter titel skrivet på ett sådant sätt att om en användare anger ett mellanslag istället för ett nyckelord kommer absolut alla inlägg att visas, vilket är fel. Jag måste ändra det reguljära uttrycket eller förbjuda användaren att ange ett mellanslag istället för en sökord. Dessutom har min dator och min lärares dator olika datum- och tidsformat och jag insåg att jag hade skrivit fel instruktioner för den användare som ville ändra datumet för sitt inlägg, eftersom jag hade skrivit en mall enligt mitt datum- och tidsformat, men för min lärare var mallen redan fel.

För att undvika olika fel arbetar jag med felhantering:

1. jag använder TryParse
2. jag skriver olika villkor, t.ex. att kontrollera att listan inte är tom om användaren vill göra något med den
3. istället för att skriva ett specifikt tal för en for-loop skriver jag list.Count eller använder foreach –loop
4. jag tänker på vilka misstag användaren kan göra och berättar vad som går fel och hur det ska åtgärdas
5. jag följer kompilatorns instruktioner, som talar om för mig om det finns ett syntax- eller kompileringsfel i min kod. Till exempel, när jag skriver en metod, kontrollerar jag att den datatyp jag anger matchar det värde som metoden returnerar. Ser jag till att alla nödvändiga parametrar tillhandahålls för att använda metoden
6. jag konverterar värdena för vissa variabler så att det inte uppstår fel på grund av inkompatibilitet mellan datatyper
7. jag ser till att variabels livslängd är så kort som möjligt och att inga fel uppstår på grund av att en variabel inte existerar i ett visst sammanhang.

Jag skrev en metod för varje menyval. På så sätt ser programkoden mer läsbar och begriplig ut. Vissa metoder kan användas flera gånger, vilket gör att man slipper upprepa vissa delar av koden. Jag skrev ingen klass eftersom det var förbjudet i uppgiftsbeskrivningen.

# **Hur har jag arbetat med den linjära sökningsalgoritmen**

Jag använde pseudokod och beskrivningen från läroboken (sid 156-159) för att arbeta med linjär sökning. För att användaren ska hitta en titel som består av flera ord och innehåller ett sökord använde jag mig av ett reguljärt uttryck. För att jämföra det reguljära uttrycket med inläggstitlar använde jag Regex.IsMatch(titel, sökord). Jag skrev också att programmet skulle ignorera om sökordet och inläggets titel är skrivna med stora eller små bokstäver (RegexOptions.IgnoreCase). Jag använde inte en vanlig jämförelse (titel == sökord) eftersom man i så fall inte kan hitta en inläggstitel som består av flera ord genom att bara ange ett ord.

Programmets sökning kunna göras bättre igenom att använda mer avancerad algoritm som binär sökning. I mitt program kan man söka igenom titlar som innehåller mer än ett ord. Det också gör sökning bättre. I mitt program kan menyn för att välja sökningstyp (efter namn eller efter id) göras tydligare genom att ange siffror istället för ord som menyval.

# **Hur har jag använt for och/eller foreach-loopar**

Jag har använt for och foreach-loopar f ör att arbeta med list och vektor. För att visa alla blogg inlägg använde jag foreach-loop som gör att man kan bearbeta enskilda inlägg från en lista med inlägg [minBlog]. För att ändra eller ta bort ett inlägg använde jag for-loop, för att söka efter ett inlägg med dess id. Att använda for-loop är bekvämt i det här fallet, eftersom man behöver göra lika många loopar som det finns inlägg i bloggen. For-loopen behövs också för sökning och sortering.

# **Hur hanterade jag en struktur med en lista som innehåller vektorer**

För att arbeta med en lista som innehåller vektorer är for- och foreach-looparna väl lämpade för att hjälpa att bearbeta enskilda element, men det sker automatiskt. Man måste vara noga med hur man anger index för ett element och ibland kan det vara en utmaning att hitta ett sätt att få rätt index. Man måste också fundera på var man ska deklarera och initiera listan och vektorn, så att man inte stöter på ett kompileringsproblem när man skriver kod där variabeln inte finns i det sammanhanget. Jag placerade listan [minBlog] före while- loopen och vektorn [minInlägg] inuti while- loopen, men inte inuti switch-struktur. Att sortera och söka igenom listan och vektorer visade sig också vara en intressant uppgift. Jag hanterade de med hjälp av CompareTo() metoden att jämföra elementen av listan med varandra.

# **Analys**

Jag har försökt tänka på möjliga fel och informera användaren om dem. Jag har försökt göra programmet enklare för användaren. T. ex. när jag arbetar med den inre menyn har jag skrivit en extra loop som, när fel uppstår, gör att användaren stannar kvar i den inre menyn i stället för att börja om från huvudmenyn. Men det visade sig ändå att mina instruktioner till användaren inte alltid är tydliga. Jag t. ex. har inte heller tänkte om eventuella skillnader i datum- och tidsformat.

Man skulle kunna använda metoder och klasser på ett effektivt sätt i mitt program. Jag ska beskriva några exemplar. Väldigt ofta behöver man använda sortering i det här programmet, det går nog bekvämare att skriva en generisk metod som kan användas som en komponent, t.ex. när man gör en sökning. Även vissa andra handlingar kan separeras som metoder, till exempel TryParse kombinerat med att kontrollera om användaren har angett rätt värde. Man måste hela tiden kontrollera om listan är inte tom, den kan också skrivas som en metod och återanvändas. Därför måste man tänka mer noggrant på vilka delar av koden som behöver avdelas och skrivas som fristående metoder.

Bloggen kan representeras som en klass, där klassmedlemmarna skulle vara titel, text, id, datum. Detta kan vara mer effektivt, eftersom för varje klassmedlem man kan välja en mer lämplig datatyp. Nu måste jag använda konvertering, eftersom dess strängvektor kan innehålla bara strängar, men sortering efter datum t.ex. blir felaktig om man lämnar datum och tid som sträng.

Det finns något annat i programmet som skulle kunna förbättras. T.ex. skulle jag vilja förbättra sättet att ändra inlägg. Just nu måste användaren skriva om det nya innehållet från början för att ändra en del, som en titel, en text eller ett datum. Om min text till exempel består av 10 ord, men jag bara vill ändra 1 ord, måste jag ändå skriva om alla 10 orden igen. Mer effektivt är när användaren kan ändra texten som i en textredigerare genom att flytta markören till önskad plats i texten.

# **Reflektion**

Om jag skulle göra om arbetet från början, skulle jag vara ännu mer noggrann i min planering för att avdela fler metoder. Jag skulle ha tänkt bättre på instruktioner för användaren. Jag skulle ha testat mitt program på andra människor innan jag visade det för läraren.

# Med de verktyg vi har lärt oss använda, vilka andra sorters program skulle man kunna göra:

1. en slumptalsgenerator
2. ett program som låter dig översätta tid och datum till ett annat format
3. ett enkelt dataspel
4. olika typer av menyer, till exempel innehållsförteckningen i en e-bok eller en restaurangmeny, eller en bruksanvisning
5. en primitiv kalkylator
6. en poäng-, mål- röst och så vidare räknare
7. anteckningar
8. formulär för att fylla i användarens information på webbplatsen
9. beräkning av rabatt
10. valutakonverterare
11. datumräknare: ett verktyg som räknar ut skillnaden mellan två datum eller lägger till/subtraherar dagar från ett visst datum
12. ett program som analyserar en text och räknar frekvensen av varje ord
13. ett program som utför olika textomvandlingar, som att ändra alla bokstäver till versaler eller gemener, eller att hitta och ersätta ord.

# Vad behöver jag lära mig för att kunna förbättra programmet vidare:

1. klasser och objektorienterad programmering
2. olika bibliotek och metoder som de erbjuder
3. tekniker för att göra ett program mer användarvänligt
4. hur kan man förbättra söknings- och sorteringsalgoritm
5. felhantering
6. hur kan man jobba med DateTime
7. hur kan man jobba med text