Uppgift Ordspelet

I denna uppgift ska du skapa ett spel som användaren ska kunna spela mot datorn. Läs igenom och se till att du förstår hela uppgiftsbeskrivningen innan du börjar utveckla programmet.

Spelet går ut på att spelare A tänker på ett ord och spelare B ska försöka räkna ut vilket ord A tänker på genom att gissa på ord och därigenom få ledtrådar. Dessa ledtrådar består av hur många bokstäver som är rätt *och* på rätt plats samt hur många bokstäver som är rätt *men* är på fel plats. Notera att bara antalet korrekta bokstäver ska anges till spelare B, inte vilka bokstäver som avses. Ordet som spelare A tänker på måste vara ett svenskt ord med 5 bokstäver utan upprepade bokstäver och som inte är ett egennamn. Ordet får inte vara en tempus- eller pluralböjning. Ord som spelare B gissar på behöver inte följa dessa krav. Spelare B kan alltså gissa på **ABCAB** om hen så vill. Stora och små bokstäver ska behandlas på samma sätt, spelet ska alltså *inte* vara case sensitive.

Ett exempel på en spelomgång ges nedan.

1. Spelare A tänker på ordet **Polis**.
2. Spelare B gissar på **Extas**.
3. Spelare A svarar att en bokstav är rätt *och* på rätt plats.
4. Spelare B gissar på **Pirat**.
5. Spelare A svarar att en bokstav är rätt *och* på rätt plats och att en bokstav är rätt *men* på fel plats.
6. Spelare B gissar på **Polis**.
7. Spelare A svarar att det var rätt.

Notera att spelare A i steg 5 menar att ‘’**p**’’ är på rätt plats och ‘’**i**’’ ska vara med men är på fel plats. Bokstäver som är rätt och på rätt plats ska alltså inte räknas till bokstäver som är rätt men på fel plats. Varje bokstav kan alltså bara höra till en av kategorierna.

Till din hjälp att lösa uppgiften finns filen **words.txt** där 1862 ord som möter kraven ovan är angivna.

Del 1 - Programmet vs. Användaren

Du ska i denna del utveckla spelet så att användaren tar rollen att gissa ord samtidigt som det blir datorns uppgift att *tänka* på ett ord och ge ledtrådar till användaren. En schematisk skiss av programmet kan se ut som följer.

1. Programmet läser in filen **words.txt**.
2. Programmet väljer slumpmässigt ett av orden i **words.txt**.
3. Användaren matar in en gissning till programmet.
4. Programmet analyserar användarens gissning.
    a. Om användaren har gissat rätt ska antalet gissningar skrivas ut och programmet avslutas.
    b. Om användaren ger upp ska det rätta ordet skrivas ut och programmet avslutas.

    c. Om användarens gissning är ogiltig, t.ex. om gissningen innehåller fler eller färre tecken än 5 ska gissningen inte räknas utan programmet ska hoppa till steg 3.
    d. I annat fall svarar programmet hur många bokstäver som var rätt och på rätt plats och hur många som var rätt men på fel plats. Sedan återgår programmet till steg 3.

Ovanstående är som sagt en schematisk skiss av programmet. Försök att utöka skissen ytterligare genom att besvara frågorna, vilka funktioner eller metoder behövs? Vilka delar av programmet har ett gemensamt ansvarsområde? Hur ska in- och utmatning gå till?

Tips: Du kan tills vidare strunta i att utveckla ett GUI och istället skapa ett enkelt command line interface (CLI) för att fokusera på grundfunktionerna i programmet. Du bör dock designa ditt program på ett sådant sätt att CLI:et är lätt att byta ut mot ett GUI i Del 3.
Tips: För att testa programmets logik, till exempel i 4.d, kan det vara smart att tillfälligt *fuska*. Till exempel genom att programmet skriver ut vilket ord den tänker på efter steg 2 eller att ordet hårdkodas in. Var dock noga med att detta återställs efter projektet.

Del 2 - Användaren vs. Programmet

I denna del ska rollerna vara de ombytta gentemot i Del 1. Nu ska användaren tänka på ett ord och det är datorns uppgift att lista ut vilket. En schematisk skiss över programmet kan lyda.

1. Programmet läser in filen **words.txt**.
2. Programmet gissar på ett slumpvis ord i sin lista med potentiella ord.
3. Användaren matar in hur många bokstäver som var rätt och på rätt plats samt hur många som var rätt men på fel plats. Exempelvis genom att låta två siffror, åtskilda av mellanslag, representera rätt plats respektive rätt bokstav men på fel plats.
4. Programmet läser uppdaterar sin lista med potentiella ord.
5. Återgå till steg 2 om gissningen inte var rätt och om programmet inte har uttömt sitt vokabulär. Om programmet inte har fler ord att gissa på ska de ge upp.

Du måste besvara följande frågor när du utvecklar denna del.

- Hur ska in- och utmatning gå till? Alltså hur användaren ska rätta programmets gissning.
- Hur ska programmet välja det ord det ska gissa på och hur ska den begränsa sin sökrymd under spelets gång?

Tänk återigen igenom vilka funktioner eller metoder som behövs. Vid det här laget bör du arbeta objektorienterat genom att skapa klasser med distinkta ansvarsområden.

Del 3 - Grafiskt gränssnitt

Skapa nu ett grafiskt gränssnitt (GUI) till ditt program. Det ska finnas någon form av huvudmeny där användaren kan välja att spela enligt varianten i Del 1 eller 2.
Du får själv välja hur den grafiska layouten ska se ut. Till exempel om alla programmets delar ska ske i samma fönster eller om det kanske skapas ett nytt fönster per spel. Ska layouten till spelvarianterna i Del 1 och 2 se likadana ut eller kräver de olika layout? Vilka widgets bör användas för att interagera med programmet? Du bör börja med att skissa en

mockup med penna och papper. Utifrån den skissen får du en klarare uppfattning om vilka widgets som behövs och vad det ska användas till. Dessutom kan du dra pilar mellan olika vyer av ditt program eller från input-widgets, som till exempel knappar till de callback-funktioner som behövs.

Tips: Försök att inte blanda GUI-funktionalitet och spelets underliggande logik alltför mycket. Skapa istället en eller flera GUI-klasser vars enda uppgift är att positionera och interagera med widgets. Dessa klasser kan innehålla de callbackmetoder som behövs men bör använda sig av instanser av andra klasser för att sköta spelets underliggande logik som utvecklades i Del 1 och 2.

Del 4 - Utökningar

Om du har kommit så här långt, och gjort det på ett bra sätt, så har du klarat av den grundläggande uppgiften.
Men för att lära dig mer och öka dina chanser för ett bra betyg så bör du vidareutveckla ditt program på något vis. Det finns många sätt att göra detta, och du väljer själv vilka du tycker är rimligt att prioritera. Här är några förslag:

- Om programmet inte längre har något ord att gissa på ska programmet be användaren att mata in de ordet hen tänkte på. Då ska programmet kontrollera att ordet är giltigt, alltså inte har upprepade bokstäver. Sedan ska programmet gå tillbaka och kontrollera så att användaren har gett korrekta ledtrådar, om användaren har gjort fel ska detta påtalas. Om användaren inte har gjort något fel ska programmet fråga om ordet ska läggas till i dess vokabulär, d.v.s. läggas till i **words.txt**.
- Gör så att programmet sparar statistik på tidigare spel och presenterar detta i en *high score*-tabell och medelvärde på antalet gissningar.
- Gör så att användaren kan spara ett pågående spel för att sedan återuppta vid ett senare tillfälle.
- Lägg till funktionen att programmet kan visa upp sina kvarvarande potentiella ord när det spelas enligt Del 2.
- Försök att göra så att ditt program aldrig behöver fler än 6 gissningar. Eller optimera ditt program så mycket som möjligt och beskriv hur du tänker. Finns det bättre och sämre ord att gissa på?
- Gör så att eventuella buggar i programmet som ger upphov till errors gör så att en informationsruta dyker upp där felet beskrivs. Alltså, istället för att skriva ut ett errors stack-trace i terminalen så ska felet visas i en pop-up som tillhör GUI:t.
- Lägg till möjligheten att konfigurera GUI:t i en inställningsmeny. Till exempel möjligheten att ändra bakgrundsfärg, typsnitt och liknande. Detta bör också kunna sparas så att programmet *kommer ihåg* sin senaste inställning nästa gång det körs.
- Gör en tredje variant av spelet där programmet både gissar och svarar. Den delen av programmet som gissar får såklart inte tillgång till det tänkta ordet. Det ska gå att hänga med i spelet för användaren till exempel genom att se historik under spelet.