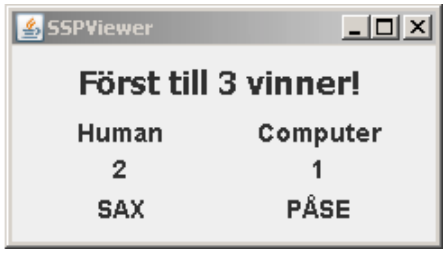
Programmeringsuppgift 3

Avsikt

Avsikten med programmeringsuppgiften är att du ska träna på att skriva ett fullständigt grafiskt program. Samtidigt ska du öva på att använda ett antal klasser i samma program.



Inlämning

Din lösning av uppgiften lämnas in via It's learning *senast kl 09.00 torsdagen den 29/10*. Du ska placera *samtliga källkodsfiler i paketet p3* i en zip-fil.

Zip-filen ska du ge namnet AAABBBP3.zip där AAA är de tre första bokstäverna i ditt efternamn och BBB är de tre första bokstäverna i ditt förnamn. Använd endast tecknen a-z när du namnger filen.

- Om Rolf Axelsson ska lämna in sina lösningar ska filen heta AxeRolP3.zip.
- Om Örjan Märla ska lämna in sina lösningar ska filen heta MarOrjP3.zip.
- Är ditt förnamn eller efternamn kortare än tre bokstäver så ta med de bokstäver som är i namnet: Janet Ek lämnar in filen EkJanP3.zip

Granskning

Din granskning ska omfatta 1-2 A4-sidor. Granskningen ska vara som pdf-dokument och *lämnas in via It's Learning innan du redovisar fredagen den 30/10*. Namnet på granskningen ska vara samma som zip-filen, dvs. *AAABBBP3.pdf*.

Granskning av Programmeringsuppgift 2

Lösning inlämnad av Eva Lind

Granskare: Einar Bok

Datum: 29/10-2015

Funktion, lösning:
Hur är funktionen i programmet som helhet och hur är funktionen i de olika delarna.
Hur bra är lösningen?

Indentering, metodnamn, variabelnamn mm:
Hur välskrivna är klasserna? Är identifierarna väl valda?

Kommentarer:
Är klasser, konstruktorer och metoder väl dokumenterade. Finns det delar i koden som borde varit kommenterade?

Redovisning

Redovisning sker *fredagen den 30/10*. Redovisningstid publiceras på It´s learning under *torsdagen den 29/10*. Kom väl förberedd till redovisningen. Kom i god tid till redovisningen så du är beredd då det är din tur. Se till att du är inloggad på en dator (eller har egen dator), att eclipse är igång på datorn och att det går att exekvera dina lösningar.

En redovisning sker genom att:

- Studentens lösningar körs.
- Granskaren redogör för sina bedömningar
- Studenten svarar för sina lösningar
- Labhandledaren ställer kompletterande frågor
- De studenter i gruppen som inte redovisar är åhörare.

Godkänd uppgift signeras av läraren på lämpligt papper, t.ex. Redovisade uppgifter (se kurssidan). Du ska spara den signerade utskriften tills kursen är avslutad.

Om labhandledaren anser att det endast krävs *mindre komplettering för att lösningen ska godkännas* kan denna komplettering äga rum direkt efter redovisningen. Labhandledaren granskar kompletterad lösning då tiden medger.

Om labhandledaren anser att det endast krävs *mindre komplettering för att granskningen ska godkännas* kan denna komplettering äga rum direkt efter redovisningen. Labhandledaren granskar kompletterad granskning då tiden medger.

Uppgift

Programmeringsuppgiften går ut på att konstruera ett ”Sten, sax och påse-spel”. På kurssidan finns filen ***SSP.exe*** vilket visar hur ett Sten, sax och påse-spel kan fungera. Testspela gärna spelet så du förstår vad du ska göra. I slutet av denna uppgift finner du dessutom regler för spelet.

OBS! Din lösning ska visa informtion till spelaren i ett fönster och hantera spelarens klick på knappar i ett annat fönster. Som i figurerna nedan.

SSPViewer innehåller sju st *JLabel*. *SSPUserInput* innehåller fem st *JButton*. Händelser är endast kopplade till knapparna i *SSPUserInput*.

I exempeln nedan används endast text för att t.ex. visa spelarens val. Det är naturligtvis fullt tillåtet att använda bilder i stället för texter, lämpliga ljud mm. Allt för att göra programmet trevligare.

När spelaren eller datorn fått 3 poäng avslutas spelet. Val-knapparna dimmas om du löst extra-uppgiften.

När användaren klickat på "Nytt spel" är *SPPViewer* "noll-ställd". Samtliga knappar är aktiva.

Användaren valde STEN och dator-spelaren valde PÅSE. Dator-spelaren får ett poäng.

Du får gärna använda bilder i din lösning! Och ljud!

Krav på design av programmet

Ditt program ska innehålla minst fyra klasser, nämligen

SSPViewer – Visar spel-information.

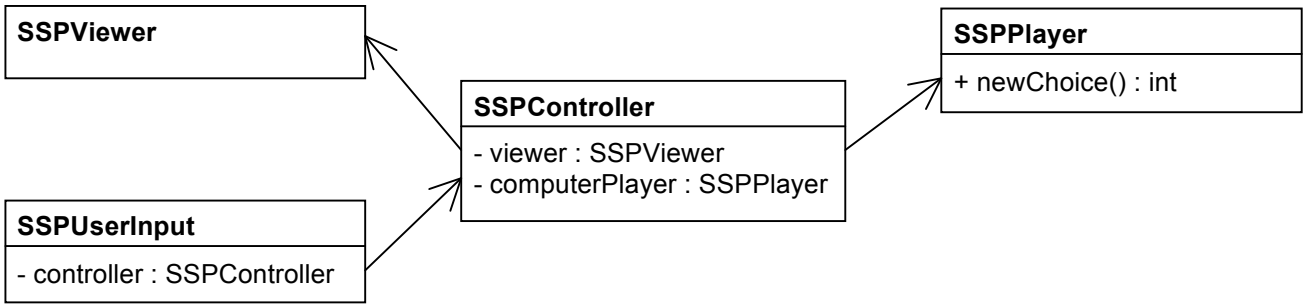
SSPUserInput – Låter användaren välja **STEN/SAX/PÅSE** eller **Nytt spel** eller att **Avsluta** programmet.

SSPPlayer – En klass som sköter datorspelaren. Datorspelaren ska endast leverera nya val till *SSPController*.

SSPController – Sköter spelet, t.ex.
* ordnar med nytt spel (vid klick på Nytt spel)
* jämför spelarens val med datorns val och delar ut poäng
* avsluta spelet då en spelare nått 3 poäng
* avslutar programmet (vid klick på Avsluta).

Nedan visas delar av klassdiagram (klassnamn + vissa instansvariabler) över klasser som måste ingå i din lösning. En pil från en klass till en annan klass visar att den första klassen känner till den andra klassen. Detta återspeglas också av de instansvariabler som finns i klassen. Det kommer att vara nödvändigt att lägga till instansvariabler i flera klasser.

Metoden *newChoice* i *SSPPlayer* är endast ett förslag på metod som bör finnas i klassen.



Ditt program ska startas med nedanstående main-metod:

```
public static void main( String[] args ) {
    SSPPlayer player = new SSPPlayer();
    SSPViewer viewer = new SSPViewer();
    SSPController controller = new SSPController(player, viewer);
    SSPUserInput userInput = new SSPUserInput(controller);

    JFrame frame1 = new JFrame( "SSPViewer" );
    frame1.setDefaultCloseOperation( JFrame.EXIT_ON_CLOSE );
    frame1.add( viewer );
    frame1.pack();
    frame1.setVisible( true );

    JFrame frame2 = new JFrame( "SSPUserInput" );
    frame2.setDefaultCloseOperation( JFrame.EXIT_ON_CLOSE );
    frame2.add( userInput );
    frame2.pack();
    frame2.setVisible( true );
}
```

Att skriva programmet

Planera noga programmet innan du börjar skriva det. Du bör fundera över vad som ska hända när användaren klickar på olika knappar.

All spellogik ska placeras i *SSPController*. Här gäller det alltså att planera vilka instansvariabler som behövs, vilka metoder som måste kunna anropas mm. Metoder som kan finnas i klassen är *newGame*, *finish* och *newChoice*. Dessa metoder anropas från *SSPUserInput* när användaren klickar på de olika knapparna.
Vad ska göras om användaren klickar på ”Nytt spel”?
Vad ska göras om en av spelarna nått 3 poäng? Hur håller man reda på spelarnas poäng?
Vad ska göras då människo-spelaren gör ett val?

Hur ska valen sten, sax och påse representeras i programmet? Ett sätt är att koda dessa som heltal, t.ex. låta 0 representera STEN, 1 representera SAX och 2 representera PÅSE.

Ett speciellt problem är hur man låter datorn välja. I förslaget ovan innehåller klassen *SSPPlayer* metoden *newChoice()* vilken sköter valet. Metoden returnerar slumpmässigt något av talen 0, 1 eller 2.

Regler för Sten, sax och påse

Sten sax och påse spelas av två spelare. I det här fallet är datorn den ena spelaren. Vardera spelaren väljer sten, sax eller påse i varje omgång. Sedan jämför spelarna sina val. Följande gäller:

- Sten vinner över sax. Med det menas att om den ene spelaren valt en sten och den andre en sax så erhåller spelaren som valt sten en poäng.
- Sax vinner över påse.
- Påse vinner över sten.

Om spelarna väljer samma så får ingen av dem någon poäng i den omgången. Den spelare som först uppnår tre poäng har vunnit spelet.

Att dimma knapparna vid avslutat spel – Extra, ej obligatoriskt

En knapp kan dimmas genom anrop till setEnabled-metoden, t.ex.

```
btnStone.setEnabled( false );
```

Knappen aktiveras på nytt genom anropet

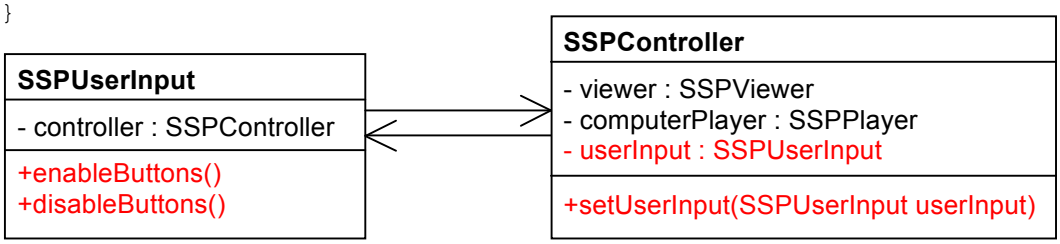
```
btnStone.setEnabled( true );
```

Men för att SSPController-objektet ska kunna styra över om knapparna är aktiva eller dimmade måste

- SSPController-objektet har en referens till SSPUserInput-objektet
- Måste SSPUserInput-klassen ha metoder (eller metod) för att aktivera + avaktivera knapparna

Ett sätt att hantera detta är att SSPUserInput-objektet anropar en set-metod i SSPController-objektet (setUserInput) från sin konstruktorn, och som argument använder en referens till sig själv.

```
public SSPUserInput(SSPController controller) {
    this.controller = controller;
    this.controller.setUserInput( this );
    // övrig kod i konstruktorn
}
```



Klassdiagrammen visar endast utvalda instansvariabler och metoder.