Laboration 4 – eget spel

Mål: Du ska skriva ett eget program. Du ska träna på att deklarera och använda variabler, att göra anrop mellan klasser samt att använda alternativ och repetition.

Förberedelseuppgifter

- Läs igenom texten i avsnittet Bakgrund.
- Hitta på ett spel och bestäm hur det ska fungera.
- Skissa ditt program (gärna pseudokod) på papper.
- Skapa klasserna ditt program ska innehålla (enligt uppgift 2 under datorarbete nedan)
- Börja skriva kod så att du (minst) har ett minimalt program som går att köra när laborationen börjar. Men du får givetvis gärna gör mer, för att säkert hinna klart!

Bakgrund

På den här laborationen ska du skriva ett textbaserat spel. Du ska själv hitta på hur det ska fungera. Endast fantasin sätter gränser, det måste inte vara ett spel som redan finns. Vill du skriva något annat program, som inte är ett spel, så går det också bra. Stäm av med din lärare eller handledare om du är osäker. Känns det begränsat med textbaserat? Det går att göra mer än vad man tror. Googla gärna ”textspel” för inspiration.

Krav på programmet

- Utveckla ditt program stegvis. Börja med ett program som bara gör något litet. Då har du tidigt ett program som fungerar. Bygg sedan ut det efterhand.
- Ditt program ska bestå av minst två klasser. Det innebär minst en klass innehållande (minst) en main-metod och en annan klass med flera metoder. Klassen med main-metoden ska skapa och använda ett objekt av den andra klassen för att köra spelet.
- Ditt program ska vara uppdelat i flera metoder. Din main-metod ska anropa minst en annan metod för att dra igång spelet, därefter ska relevanta metoder användas för att köra olika delar av spelet.
- Ditt program måste innehålla minst ett alternativ (`if-sats`) och minst en repetition (`for`- eller `while`-sats).
- Din kod ska vara lätt att läsa och förstå (bra val av klass- och variabelnamn, vettig indentering etc).
- Det är en konst att begränsa sig. Se till att du inte tar dig vatten över huvudet. Kontrollera gärna din idé med labhandledaren.

Inspiration

Här finns några förslag att hämta inspiration ifrån. Du kan använda något av förslagen, modifiera något av förslagen eller hitta på något helt eget spel.

- Spela ”gissa talet” och ge ledtrådar om talet är för litet eller för stort.
- Hitta på något enkelt tärningsspel att spela med användaren.
- Träna användaren i multiplikationstabellen.
- Rita många kvadrater i ett fönster och låta användaren gissa antalet.
- Kolla reaktionstiden hos användaren genom att mäta tiden det tar att trycka Enter efter att man fått vänta en slumpmässig tid på att strängen ”NU!” skrivs ut. Om man trycker Enter innan startutskriften ges blir den uppmätta tiden 0 och på så sätt kan ditt program detektera att användaren har tryckt för tidigt. Mät reaktionstiden upprepade gånger och beräkna medelvärdet.

- Låt användaren svara på flervalsfrågor om din favoritfilm.
- Spela sten/sax/påse med användaren.
- Låt användaren på tid så snabbt som möjligt skriva olika ord baklänges.
- Ett textbaserat äventyrsspel där användaren får gå igenom en värld och göra val, svara på gåtor etc.

Några användbara metoder och hjälpklasser

Här följer några tips på hur du kan jämföra strängar, dra slumptal och hantera tid i Java. Vi kräver inte att du använder alla dessa finesser, men du har troligen nytta av något av dem.

Strängjämförelse: använd metoden equals för att jämföra om två strängar är lika. **Exempel:**

```
String svar = scan.next();
if (svar.equals("nej")) {
    // ... något händer ...
}
```

Ibland vill man inte att det ska spela någon roll ifall användaren har skrivit med små eller stora bokstäver. Då kan man istället använda `equalsIgnoreCase`:

```
String svar = scan.next();
if (svar.equalsIgnoreCase("nej")) {
    // ... något händer ...
}
```

Slumptal: använd ett `Random`-objekt för att ge variabler slumpmässiga värden:

`Random`

```
/** Skapar en slumptalsgenerator. */
Random();

/** Returnerar ett slumptal mellan 0 och bound-1. */
int nextInt(int bound);

/** Returnerar ett slumptal mellan 0.0 och 1.0. */
double nextDouble();
```

Exempel:

```
Random rand = new Random();
int nbr = rand.nextInt(10);  // nbr får slumpmässigt värde mellan 0 och 9
```

Mäta tid: använd den statiska metoden `currentTimeMillis` i klassen `System` för att mäta tid.

`System`

```
/** Returnerar aktuell tid i millisekunder, sedan någon ospecificerad
    startpunkt. Differensen mellan två värden anger hur lång tid som
    förflutit. */
static long currentTimeMillis();
```

Exempel:

```
long start = System.currentTimeMillis();
// ... här görs något som vi vill mäta tiden för ...
long end = System.currentTimeMillis();
long elapsed = end - start;
```

Fördröjning: för att låta programmet vänta en viss tid kan du använda klassen `Timer`, som finns i paketet `se.cs.lth.pt.timer`. Du importerar klassen på vanligt sätt, men notera att det finns några andra Java-klasser med samma namn, så var säker på att du väljer rätt paket.[2]

`Timer`

```
/** Fördröj programmets exekvering med ett antal millisekunder. */
static void delay(long milliseconds);
```

Exempel:

```
Timer.delay(2000);  // Programmet väntar här i 2 sekunder
```

Datorarbete

1. Logga in på datorn, starta Eclipse, öppna projektet *Lab04*.

2. Skapa en fil med namnet *SmallGame.java* (eller något namn du väljer själv). För att skapa en fil ska du
   - markera projektet *Lab04*
   - klicka på New Java Class-ikonen eller högerklicka och välja New och därefter Class
   - skriva namn på klassen (t.ex. `SmallGame`)
   - klicka på Finish

   Skapa sedan ytterligare en klass, som ligger i en annan fil. I denna klass ska din main-metod ligga. Det går bra att döpa klassen till *Main.java* (eller något annat namn du själv väljer).

3. Skriv ditt program. Testa och se till att spelet fungerar som avsett. Tänk på att utveckla ditt spel i små steg, så att du kan provköra ofta.

4. Låt någon annan kursdeltagare läsa ditt program och köra spelet. Notera den återkoppling du får på programmet. På samma sätt ska du ge återkoppling på en annan kursdeltagares program.

   Exempel på frågor man kan ställa sig är:
   - Är spelet svårt eller lätt? Är det kul? Går det som användare att begripa vad man förväntas göra?
   - Är det lätt att förstå programkoden? Är det något som är särskilt klurigt? Lärde du dig något nytt genom att läsa programmet?
   - Hur ser uppdelningen i klasser ut? Vilka är de olika klassernas uppgifter?

5. Använd den återkoppling du fick för att förbättra ditt program.

Checklista

Exempel på vad du ska kunna efter laborationen:

- Skapa .java-filer och skriva enkla program i Eclipse.
- Förstå hur klasser kan interagera.
- Förstå vad som händer när metoder anropas.
- Skriva programkod med alternativ (`if`-sats).
- Skriva programkod med repetition (`for`- eller `while`-sats).

[2] Din `import`-sats ska se ut så här: `import se.lth.cs.pt.timer.Timer;`