Mittuniversitetet
MID SWEDEN UNIVERSITY

Tentamen

Java för C++-programmerare, 7,5 högskolepoäng

Distanskurs från Östersund (DSV)
2018-01-19, kl. 8.00 – 20.00 (12 timmar)

Antal uppgifter: 5

Betygskala[1]:

F	Fx	E	D	C	B	A
Ej körbar lösning	Delvis löst uppgift 1	Godkänd lösning uppgift 1	Godkänd lösning uppgift 2	Godkänd lösning uppgift 3	Godkänd lösning uppgift 4	Godkänd lösning uppgift 5

Betyget E-A innebär godkänt, Fx och F är underkänt. För betyget E krävs en godkänd och körbar lösning på uppgift 1. För betyget D krävs en godkänd och körbar lösning på uppgift 1 och 2 o.s.v.

Anvisningar:

Förutom de regler som specificerats på sidan Tentaregler i Moodle för denna kurs gäller även följande:

- Tentamen ska skrivas personligen. Det är inte tillåtet att kopiera kod från någon annan, från webbsidor eller liknanden. Det är inte heller tillåtet att ta hjälp från någon annan person.
- Det är tillåtet att använda litteratur, API:er, egen kod från laborationer och att söka information på webben.
- Innan skrivningstiden är slut måste du skicka in ditt svar. Detta görs i Moodle på samma sätt och på samma ställe där du tidigare har lämnat in dina inlämningsuppgifter.
- Det som ska lämnas in är zip-fil innehållandes alla filer du skrivit för att lösa uppgifterna (inga .class filer). Har du använt Eclipse för att lösa uppgifterna packar du hela projektet i en zip-fil. Döp zip-filen enligt följande format:
  ååmmdd-nnnn, Efternamn Förnamn.zip
- Ta en titt lite nu och då i forumet i Moodle eftersom jag där kommer att skriva eventuella tips/förtydligande som behövs för uppgifterna.
- Tentamen får skrivas på den plats du angav i din anmälan.
- Om det blir några problem med din nätanslutning under tentamenstiden, eller när du ska lämna in ditt svar, måste du omedelbart kontakta mig, Robert Jonsson på telefon 010 - 1428138. Jag kan även nås via forumet och meddelande i Moodle, samt på robert.jonsson@miun.se.

Inledning

Läs igenom alla uppgifter innan du sätter i gång med att skriva kod. Siktar du på ett av de högre betygen måste du kanske tänka på detta redan i koden för den tidigare uppgifterna. Tänk på att inte ”baka in” all kod i en och samma klass utan dela upp koden i lämpliga klasser och metoder. För varje ny uppgift du påbörjar är det lämpligt att kopiera hela föregående uppgift till en ny mapp så att föregående uppgift finns kvar i oförändrat skick. Skicka bara in lösningen för det betyg du siktar på. Återanvänd eller bygg gärna vidare på kod från dina inlämningsuppgifter eller exempel i kursboken. Skriv en kommentar i samband med inlämningen i Moodle om vilket högsta betyg/uppgift du försökt med.

I uppgifter där ett GUI används är det inte tillåtet att meddela användaren om vad som händer i applikationen med System.out.print (debug-utskrifter är dock tillåtna). I ett GUI är det inte heller tillåtet att hämta indata från användaren med t.ex. Scanner eller System.in. Funderar du över något eller om något är oklart vad gäller tentamens-uppgifterna är det bara att höra av dig. Antingen i kursens forum eller via e-post till kursansvarig.

Gör du mindre missar och fel i en uppgift behöver inte betyda att uppgiften underkänns. Om däremot flera uppgifter innehåller fel och missar kan betyget komma att sänkas ett eller flera steg. Om du t.ex. har lämnat in en lösning för uppgift 4 (betyg B) och uppgift 1 och 3 innehåller några missar, kan ditt betyg komma att sättas till C.

Dokumentationskommentarer behöver inte skrivas om det inte uttryckligen står så i någon uppgift. Kommentera i övrigt din kod så som du själv anser det vara nödvändigt.

Observera att eventuella klassdiagram som används i uppgifterna inte nödvändigtvis är helt kompletta/korrekta utan kan sakna vissa detaljer. Detta för att inte avslöja allt för mycket hur översättningen till Javakod ska göras.

Tentan handlar denna gång om spelet Bingo.

Uppgift 1 (betyg E)

Bingo är ett spel där en utropare slumpvist ropar ut ett namn (Bertil, Ivar, Olof, Niklas eller Gustav) följt av en siffra (1 – 75), varpå deltagaren fyller i siffran på sin bricka om numret även finns på brickan. När en hel rad har blivit markerad - lodrätt, vågrätt eller diagonalt - har spelaren fått bingo.

Varje bricka består av fem vertikala kolumner märkta med B, I, N, G och O. Kolumn B innehåller nummer mellan 1 och 15, kolumn I innehåller nummer mellan 16 och 30 och så vidare. En klassisk bingobricka innehåller 25 nummer (fem kolumner och fem rader) och kan se ut som den främre brickan i bilden nedan. Det finns även en variant där brickan innehåller alla 75 nummer uppdelade i 3 grupper om 25 nummer (se den bakre brickan i figuren nedan). Numren är slumpvist utplacerade i varje kolumn och varje nummer finns maximalt bara en gång per bricka.

I första uppgiften ska du skapa en egen grafisk komponent (klass som ärver `JComponent` eller `JPanel`) som ska fungera som ”utropare” i bingospelet. Komponenten ska för användaren presentera det bingonummer som dras (slumpas fram). I komponenten ska minst de tre senaste dragna bingonumren synas samtidigt (gärna fler, till exempel genom att kunna scrolla fram och tillbaka). Komponenten ska skrivas i en klass som heter `BingoCaller`.

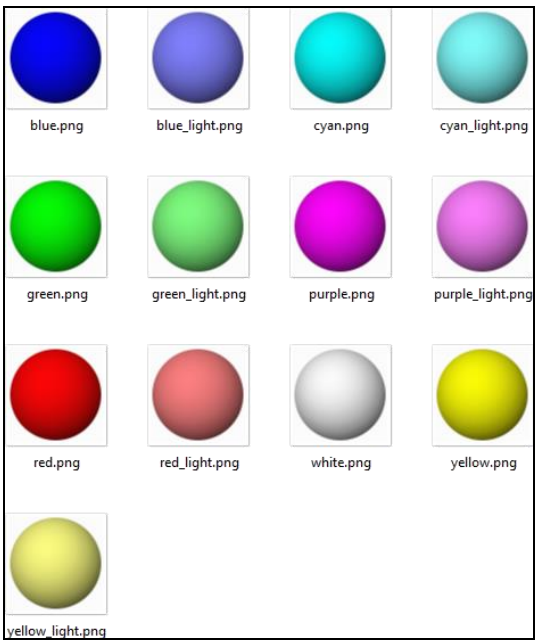
Med bingonummer menas följande:

Nummer 1 – 15 ger bingonummer B1, B2, B3, …, B15.
Nummer 16 – 30 ger bingonummer I16, I17, I18, …, I30.
Nummer 31 – 45 ger bingonummer N31, N32, N33, …, N45.
Nummer 46 – 60 ger bingonummer G46, G47, G48, …, G60.
Nummer 61 – 75 ger bingonummer O61, O62, O63, …, O75.

Om nummer 7 dras ska din `BingoCaller` med andra ord visa B7. Om nummer 55 dras ska komponenten visa G55. Och så vidare. Bingonumret ska representeras av en klass som heter `BingoBall`.

I denna uppgift behöver inte komponenten `BingoCaller` kunna slumpa nummer med mera utan det räcker att den kan visa bingonummer som du lägger in direkt i din kod.

Välj själv utformningen på `BingoBall`. Bifogat med tentan finner du två zip-filer som innehåller bilder på bollar du ska använda i din lösning (se figur 1 och figur 2). Ovanpå bollen ska sen bingonumret skrivas ut.



Figur 1

Figur 2

Skapa en klass `BingoGUI` som ärver `JFrame` och som lägger till komponenten överst i fönstret och lägg sen till minst fem bingonummer. I kod kan det t.ex. se ut så här:

```
BingoCaller caller = new BingoCaller();
caller.addBingoBall(new BingoBall(1));
caller.addBingoBall(new BingoBall(31));
caller.addBingoBall(new BingoBall(41));
caller.addBingoBall(new BingoBall(61));
caller.addBingoBall(new BingoBall (18));
add(caller, BorderLayout.NORTH);
```

Resultatet av koden ovan skulle kunna se ut som i figur 3.

Figur 3

Dela upp din kod i lämpliga metoder (och klasser) utöver vad som specificerats ovan. Den klass med vilken man startar applikationen döper du till `Bingo`. Övriga klasser som minst ska finnas med i din lösning är `BingoBall`, `BingoCaller`, `BingoGUI` och `Bingo`.

När du är klar skapar du en jar-fil av alla nödvändiga filer som behövs för att köra applikationen. Jar-filen döper du till `bingo.jar` (du behöver inte skapa en jar-fil nu om du går vidare till nästa uppgift).

Uppgift 2 (betyg D)

Du ska nu göra det möjligt för användaren att skapa bingobrickor och spara dessa till en fil. Välj själv om det ska vara 25-nummersbrickor eller 75-nummersbrickor som skapas. Oavsett vilken bricka du väljer ska första raden innehålla bokstäverna B, I, N, G, O och raderna därefter fylls med slumpade nummer. ”Rita” bingobrickan med hjälp av t.ex. | och - (se exempel nedan till vänster) eller t.ex. * (se exempel nedan till höger). Givetvis får du använda andra tecken om du vill.

```
-------------------------
| B  | I  | N  | G  | O  |
-------------------------
|  6 | 18 | 31 | 50 | 73 |
-------------------------
| 14 | 29 | 38 | 59 | 66 |
-------------------------
|  5 | 20 | 40 | 51 | 72 |
-------------------------
| 13 | 25 | 44 | 46 | 71 |
-------------------------
|  1 | 16 | 42 | 54 | 62 |
-------------------------
```

```
*************************
* B  * I  * N  * G  * O  *
*************************
*  6 * 18 * 31 * 50 * 73 *
*************************
* 14 * 29 * 38 * 59 * 66 *
*************************
*  5 * 20 * 40 * 51 * 72 *
*************************
* 13 * 25 * 44 * 46 * 71 *
*************************
*  1 * 16 * 42 * 54 * 62 *
*************************
```

I din `JFrame` från föregående uppgift ska du nu lägga till ett menyalternativ som skapar och sparar en bingobricka till fil på användarens hårddisk. Tanken är att användaren ska kunna skriva ut dessa för att använda i bingospelet. Varje gång användaren väljer att skapa en bingobricka ska nya nummer slumpas fram. Välj själv hur bingobrickornas ska namnges (filnamn).

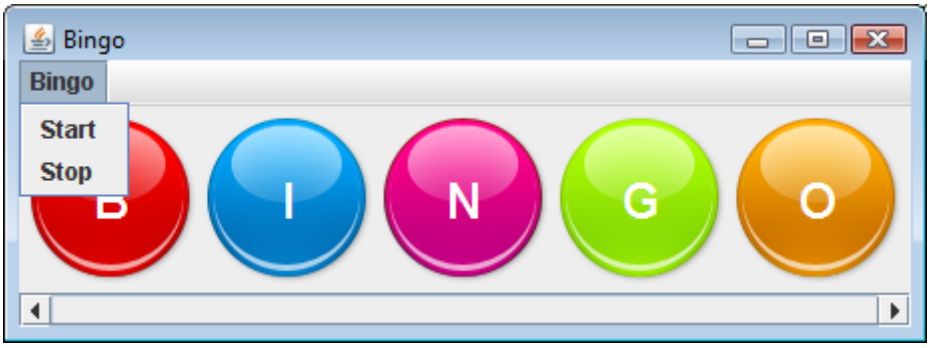
När du är klar skapar du en jar-fil av alla nödvändiga filer som behövs för att köra applikationen. Jar-filen döper du till `bingo.jar` (du behöver ännu inte skapa en jar-fil om du går vidare till nästa uppgift).

Uppgift 3 (betyg C)

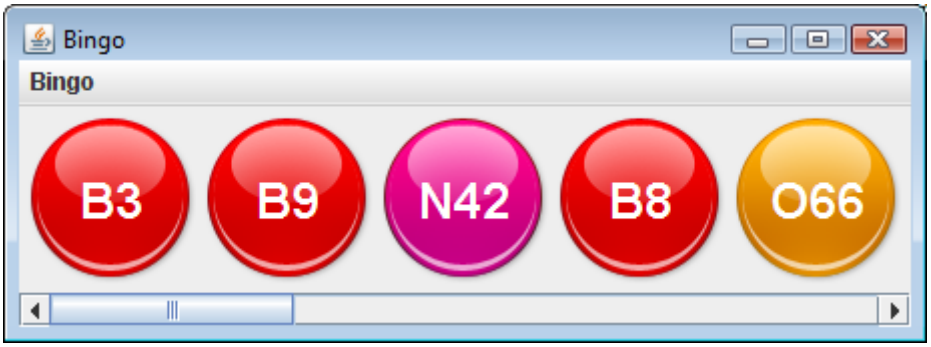
I denna uppgift ska du utöka komponenten `BingoCaller` från uppgift 1 med start- och stop-metoder. När start anropas ska första numret visas, därefter vänta en stund, visa andra numret, vänta en stund igen, visa tredje numret och så vidare till dess att alla 75 nummer har visats (eller till dess att stop-metoden anropas). Välj själv hur lång fördröjningen mellan varje dragning av nytt nummer ska vara. I den här uppgiften behöver din lösning inte kunna hantera återstarter (dvs starta dragningen igen efter att stop har valts).

Komponenten kan med fördel själv hålla reda på till exempel alla nummer, slumpa i vilken ordningen dessa ska visas, ha en metod för att visa nästa nummer etc. Ta gärna hjälp av andra klasser om du hellre vill det (läs sista uppgiften).

I din `JFrame` ska det finnas ett menyalternativ som, när användaren väljer det, börjar dragningen av alla 75 nummer samt ett menyalternativ som stoppar dragningen av numren (se exempel i figur 4 och 5).



Figur 4



Figur 5

När du är klar skapar du en jar-fil av alla nödvändiga filer som behövs för att köra applikationen. Jar-filen döper du till `bingo.jar` (du behöver inte skapa en jar-fil nu om du går vidare till nästa uppgift).

Uppgift 4 (betyg B)

Du ska nu göra en klient-server-lösning av bingospelet. Du ska skapa en server som drar bingonummer och skickar dessa till en klient som visar bingonumret i sin `BingoCaller`. Välj själv om du vill använda dig av `MulticastSocket` eller `Socket` i din lösning.

När klienten trycker på menyalternativet Start ska servern slumpa nya värden och därefter skicka bingonummer en efter en (med en fördröjning du själv väljer) ända till dess att klienten trycker på menyalternativet Stop eller till dess att servern skickat alla 75 nummer.

Servern behöver inte klara av flera samtidiga klienter i denna uppgift. Inte heller i denna uppgift behöver din lösning kunna hantera återstarter (dvs efter att klienten har tryck på stop, eller alla 75 nummer är dragna, behöver ett val av menyalternativet Start i klienten inte återuppta dragningen).

När du är klar skapar du en jar-fil av alla nödvändiga filer som behövs för att köra applikationen. Jar-filen för serverdelen döper du till `server.jar` och jar-filen för klientdelen döper du till `client.jar` (du behöver ännu inte skapa en jar-fil om du går vidare till nästa uppgift).

Uppgift 5 (betyg A)

I sista uppgiften ska du skriva om server-delen så att den klarar av flera samtidiga användare. Alla användare som ansluter till servern ska få samma bingonummer skickade till sig. När den första klienten trycker på menyalternativet Start ska servern börja skicka nummer till klienterna. Välj själv hur du väljer att hantera klienter som kommer in sent i spelet (dvs klienter som ansluter efter att dragningen av nummer har startat).

I klienten ska du ändra menyalternativet Stop till Bingo. När en användare får bingo trycker hon på menyalternativet Bingo varpå servern slutar att dra nya nummer. Server ska samtidigt meddela anslutna klienter att en användare fått bingo och att den nu kan starta ett nytt spel genom att välja menyalternativet Start. Använd t.ex. en dialogruta i klienten.

När du är klar skapar du en jar-fil av alla nödvändiga filer som behövs för att köra applikationen. Jar-filen för serverdelen döper du till `server.jar` och jar-filen för klientdelen döper du till `client.jar`.