Universitetet i Sørøst-Norge

EKSAMENSFORSIDE

Skriftlig eksamen med tilsyn

Emnekode: IA2112	Emnenavn: Objektbasert programmering videregående	
Dato: 4/6-2020	Tid fra/til: 09.00-13.00 (pluss 30 minutter til å levere)	Ant. timer: 4 timer (pluss 30 minutter til å levere)
Ansv. faglærer: Olav Dæhli		
Campus: Porsgrunn	Fakultet: Fakultet for teknologi, naturvitenskap og maritime fag (TNM)	
Antall oppgaver: 4	Antall vedlegg: 1 ark + 1 zip-fil + 1 videofil (mp4)	Ant. sider inkl. forside og vedlegg: 5

Tillatte hjelpemidler:

- Alle digitale hjelpemidler (bortsett fra chat o.l.)
- All form for kommunikasjon med andre (både digitalt og muntlig) er å anse som fusk

Opplysninger om vedlegg:

I vedlegget på side 5 vises det hvordan man kan tegne en figur (trekant, sirkel etc.) i en picturebox, hvilket er relevant for oppgave 2. Vi videoen (2,5 minutter) illustreres det hva som skal lages.

Merknader**:**

- Er det noe som er uklart eller åpenbart feil i oppgaveteksten, så skriv i besvarelsen de forutsetninger du legger til grunn for din løsning!

Leveranser:

- Et pdf-dokument der svar på teorispørsmålene, samt koden til alle klassene (Form1, Shape, Triangle osv.) og interfacet er limt inn. Start hver ny klasse (og interfacet) på en ny side.
- Levere også en zip-fil med den praktiske løsningen dere har laget

Kryss av for type eksamenspapir Ruter ☐ Linjer ☐

Oppgave 1 a (5 %)

Hvordan kan du sikre at en klasse *ikke* kan arves av andre klasser.

Oppgave 1 b (5 %)

Forklar kort hva en **Extension**-metode er og hva som er poenget med denne typen metoder.

Oppgave 1 c (5 %)

Forklar hva nøkkelordet **this** betyr/brukes til, og forklar samtidig hvorfor man *ikke* kan bruke nøkkelordet **this** til å referere til en statisk metode?

Oppgave 1 d (5 %)

Forklar kort forskjellen på **referanstyper** (reference types) og **verdityper** (value types) og gi konkrete eksempler på hver av disse typene.

Oppgave 1 e (10 %)

Forklar de viktigste likheter/forskjeller på **abstrakte klasser** og **interface**.

Oppgave 1 f (10 %)

Tenk dere en liten lokalskole som i dag gjør all karakterregistrering manuelt. I tillegg til elevene, er det ansatt en rektor, noen administrativt ansatte (deriblant eksamensansvarlige), samt lærere.

Siden all karakterregistrering ved eksamen i dag gjøres manuelt, ønsker rektor at det skal utvikles en lokal dataapplikasjon for eksamensavviklinger (inkl. karakterregistrering m.m.).

Lag ut fra et slikt «case» et bruksmønsterdiagram (**use case-diagram**), illustrert med 3 aktører og minst 4 aktuelle bruksmønstre (**use case**). Alle aktørene skal ha minst én «rolle» i diagrammet.

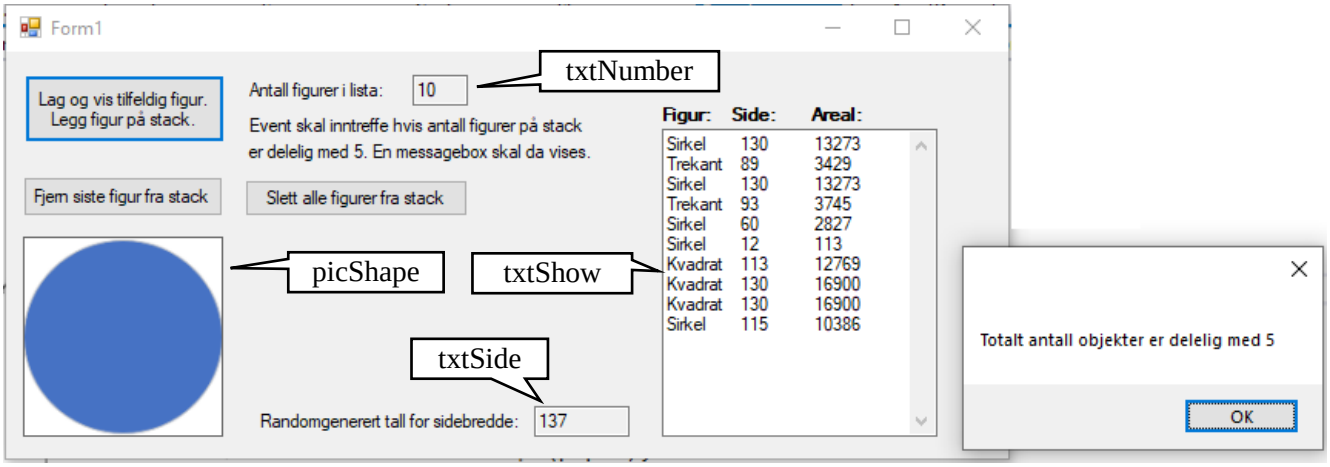
- -

Tegn dette med f.eks. **Umletino** (www.umletino.com) eller **Visio**. Alternativt kan du tegne på papir, ta bilde av det og så levere bildet som en del av besvarelsen. Velg selv norsk eller engelsk tekst.

Oppgave 2 (30 %)

I Figur 1 vises brukergrensesnittet for applikasjonen som skal lages. I **WISEflow** ligger det en videofil (**mpg**-fil) på ca 2,5 minutter, som illustrerer hvordan programmet fungerer når det er ferdig. Der ligger det i tillegg en **zip-fil** som inneholder en **C#-solution** med et grafisk brukergrensesnitt det kan tas utgangspunkt i.

Er du usikker på hvordan du tegner figurer i et grafikkområde, så se vedlegget. Et par aktuelle instruksjoner for dette er lagt inn i zip-fila, hvilket fremgår av oppgaveteksten og vedlegget.



Figur 1: Brukergrensesnitt (GUI: Graphical User Interface) under kjøring av applikasjonen som skal lages.

Spesifikasjoner for klassene Shape, Circle, Square og Triangle, samt **Interfacet IDisplayable:**

a) Programmet skal ha en **abstrakt** klasse kalt **Shape**. Denne skal ha to «properties» kalt **X** og **Y**, for å angi figurens avstand fra pictureboxens øvre venstre hjørne. I tillegg skal den ha en «**property**» kalt **Side**, som skal brukes i arealberegninger. For sistnevnte skal det legges inn en sjekk på at verdien er ikke er over **130**. Forsøkes verdien satt høyere enn **130**, skal den automatisk *settes* til **130**.

b) **Shape**-klassen skal ha to konstruktører, én parameterløs, der **X** og **Y** gis verdien **0** og **Side** gis verdien **130**, og en *med* parameter, der **X** og **Y** settes til **0**, mens verdien til **Side** angis via **parameter**.

c) **Shape**-klassen skal ha en **abstrakt** metode kalt **CalcArea**, *uten* parametere. Metoden skal returnere en verdi av typen **int**.

d) **Shape**-klassen skal ha en metode kalt **GetShapeType** som returnerer en **string** med teksten «Ukjent figurtype». Denne skal kunne overskrives av subklaser (klasser som arver fra klassen).

e) Det skal lages tre klasser kalt **Triangle**, **Circle** og **Square**, som alle skal arve fra **Shape**.

f) **GetShapeType**-metoden (som arves fra **Shape**-klassen) skal overskrives i de tre ovennevnte klassene, så de returnerer teksten "**Trekant**", "**Sirkel**" eller "**Kvadrat**".

g) Det skal lages et **Interface** kalt **IDisplayable**, som skal kobles til **Triangle**-, **Circle**- og **Square**-klassen. Dette skal inneholde følgende metode **DrawShape (Graphics drawArea)**. Interface-kravet må innfris. Se vedlegget for hvordan innholdet i metoden **DrawShape** kan lages.

h) De tre klassene **Triangle**, **Circle** og **Square** skal ha tilsvarende to konstruktører som **Shape**-klassen. Disses konstruktører skal derfor kalle opp basekonstruktørene.

i) I de tre arvede klassene kan det brukes følgende C#-formler for å beregne arealet i en metode kalt **CalcArea**: Dette skal være en metode *uten* parametere, der **Side** er en property arvet fra **Shape**:

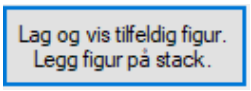
- Square-area: `area = Side * Side;`
- Circle-area: `area= (int)(Math.PI * Math.Pow(Side / 2.0,2));`
- Triangle-area: `area = (int)((((Math.Sqrt(3.0)) / 4.0) * Side * Side));`

NB! Disse formlene ligger kommentert innledningsvis i **Form1**-koden i **zip-fila**, så de kan kopieres derfra (så du dermed slipper å bruke tid på å skrive dem inn selv).

Oppgave 3 (20 %)

Oppgaver som skal løses i **Form1**:

Beskrivelse av hva som skal skje ved klikk på denne knappen:

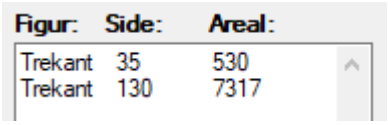


a) Det skal opprettes et tilfeldig objekt av enten typen **Circle**, **Square** eller **Triangle**. Benytt et random generert tall, 1–3, til å velge figurtype. Benytt konstruktøren med parameter for sideangivelse ved opprettelse av objektene. **Sideangivelsen** skal lages som et tilfeldig tall i området 10–200. Vis også det tilfeldig genererte tallet i tekstboksen **txtSide**. Figuren skal vises i pictureboxen **picShape**:

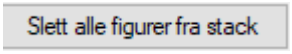
b) Hvert figurobjekt som lages skal lagres i en **Stack**-variabel (instansvariabel) kalt **shapes**, som skal inneholde elementer av datatypen **Shape**. Hvis du *ikke* vet hvordan du lager/bruker en **Stack**-variabel, kan du i stedet benytte en **List**-variabel (men det vil gi litt mindre uttelling).

c) Oppdater tekstboksen **txtNumber** med **antall** objekter **shapes**-variabelen inneholder.

d) Lag og kall opp en metode kalt **ShowAllShapes**, som viser data for alle objektene som måtte ligge i **shapes**-variabelen i en tekstboks (**txtShow**). Hver rad skal inneholde data for **figurtype**, **side** og **areal**, som vist i figuren under. Siste figur som er lagt inn, skal alltid vises øverst.

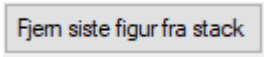


Beskrivelse av hva som skal skje ved klikk på denne knappen:



Stacken skal nullstilles/cleares, så alle elementene fjernes fra denne. Deretter må antallet i **txtNumber** oppdateres (så det blir 0) og tekstboksen **txtShow** nullstilles/cleares, så den blir uten innhold.

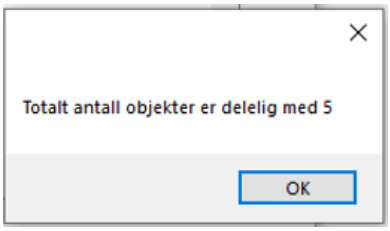
Beskrivelse av hva som skal skje ved klikk på denne knappen:



Det siste elementet som ble lagt i **shapes** (altså siste figuren som ble laget) skal fjernes fra denne. Antallet i **txtNumber** må oppdateres og innholdet i **txtShow** må oppdateres (ved å kalle opp metoden **ShowAllShapes** på nytt) og vises uten raden som er fjernet fra **shapes** (stack-variabelen).

Oppgave 4 (10 %)

e) Lag et **Event** som utløses hver gang antall objekter på stacken er delelig med 5. Da skal følgende **MessageBox**-melding poppe opp på skjermen: «Totalt antall objekter er delelig med 5».



Hvordan tegne en figur (trekant, sirkel etc.) i pictureboxen

Hva som må gjøres i figurklassene:

En metode for å tegne en figur på et **Graphics**-område, kan i figurklassene lages sånn:

```
public void DrawShape(Graphics drawArea)
{
   Bitmap pic = new Bitmap("________.png"); //Legg inn aktuell bildefil
   drawArea.Clear(Color.White); //Sletter forrige figur ved å skrive over med hvitt
   drawArea.DrawImage(pic, X, Y Side, Side);

}
```

X og **Y** er figurens plassering i grafikkområdet og er antall punkter fra øvre venstre hjørne. **Side** er i dette tilfellet benyttet både som figurens bredde og høyde.

Hva som må gjøres i Form1:

I **Form1** i zip-fila er det lagt inn deklarasjon av følgende instansvariabel: **`Graphics paper;`**

I **Form1** sin konstruktør er det lagt inn følgende instruksjon: **`paper = picShape.CreateGraphics();`** (som kobler **pictureboxens** grafikkområde til instansvariabelen **paper**

Fra **Form1** kan **DrawShape**-metoden da kalles opp sånn: **`objektnavn.DrawShape(paper);`**

Objektnavnet velger dere selv, når dere lager objekter av de tre klassene Circle, Square og Triangle.