HÖGSKOLAN
I SKÖVDE

Institutionen för informationsteknologi

TENTAMEN

Kurs: Spelprogrammering 1

Examinationsmoment: Salstentamen 3hp

Kurskod: IT148G

Datum: 2026-01-17 Tentamenstid: 9:15 - 14:30

Ansvarig lärare: Peter Sjöberg

Berörda lärare: Cecilia Wallin

Hjälpmedel/bilagor: Inga

Övrigt

Anvisningar	☐	Ta nytt blad för varje lärare
	☒	Ta nytt blad för varje ny fråga
	☒	Skriv endast på en sida av papperet.
	☒	Skriv namn och personnummer på samtliga inlämnade blad.
	☒	Numrera lösbladen löpande.
	☒	Använd inte röd penna.
	☒	Markera med kryss på omslaget vilka uppgifter som är lösta.

Poänggränser G 18-32 poäng

U 0-17 poäng

Skrivningsresultat bör offentliggöras inom 18 arbetsdagar

Lycka till!

Antal sidor totalt 6

Fråga 1 (4 poäng)

Studera koden nedan och ange vad programmet skriver ut när det körs. Programmet startar med funktionen `Program.Run`.

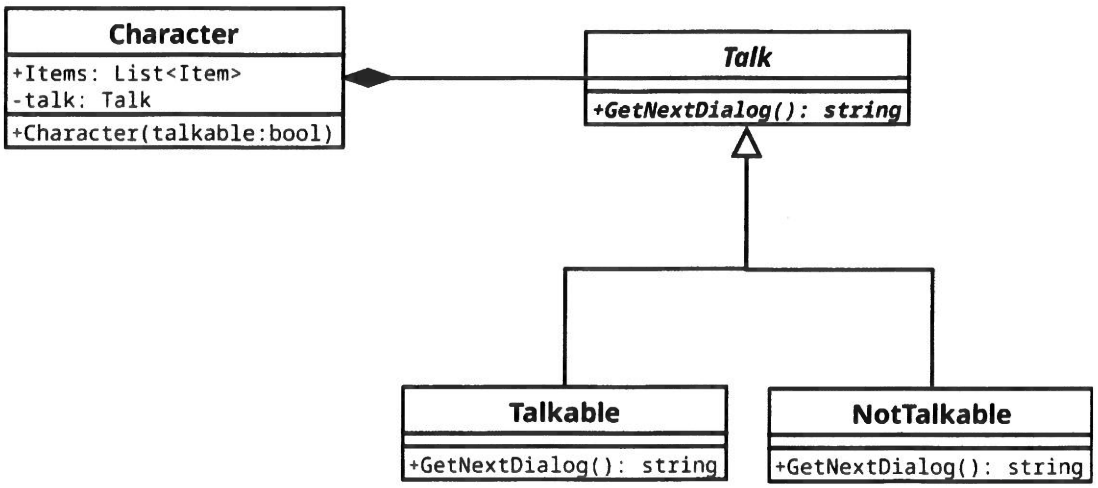
```
class Program
{
   private Furniture furniture = new Furniture();

   public void Run()
   {
      furniture = new Cabinet();
      Console.WriteLine(furniture is Cabinet);
      Cabinet cabinet = new Cabinet();
      Console.WriteLine(cabinet is Furniture);
      cabinet.Info();
      furniture.Change(1);
      furniture.Info();
   }
}


class Furniture
{
   public virtual void Info()
   {
      Console.WriteLine("Error");
   }

   public virtual void Change(int index) {}
}

class Cabinet : Furniture
{
   private Drawer[] drawers;

   public Cabinet()
   {
      drawers = new Drawer[]
      {
         new Drawer(Drawer.HandleType.Knob),
         new Drawer(Drawer.HandleType.Knob),
         new Drawer(Drawer.HandleType.Handle)
      };
   }

   public override void Info()
   {
      Console.WriteLine("Cabinet has " + drawers.Length + " drawers");
   }

   public override void Change(int index)
   {
      drawers[index] = new Drawer(Drawer.HandleType.Handle);
   }
}

class Drawer
{
   public enum HandleType
   {
      Handle, Knob
   }

   private HandleType handleType;

   public Drawer(HandleType ht)
   {
      handleType = ht;
   }
}
```

Fråga 2 (6 poäng)

Klassdiagrammet nedan visar en kodstruktur, du kan anta att trianglar kan vara ifyllda. Skriv koden som representeras av diagrammet. Funktioner ska implementeras enligt nedan.

Konstruktorn i `Character` ska skapa en instans av `Talk`, vilken variant som skapas bestäms av parametern `talkable`.

Funktionen `GetNextDialog` ska returnera en teststräng om karaktären kan prata, annars returnerar funktionen en tom sträng.



Fråga 3 (4 poäng)

När kod i C# når ett körfel (*execution error*) kommer programmet att avslutas direkt och ge ett undantag (*exception*).

a) Förklara varför programmet behöver avslutas direkt när fel uppstår enligt ovan. Använd gärna ett exempel i ditt svar.

b) Förklara hur informationen i undantaget kan användas för att felsöka koden.

Fråga 4 (4 poäng)

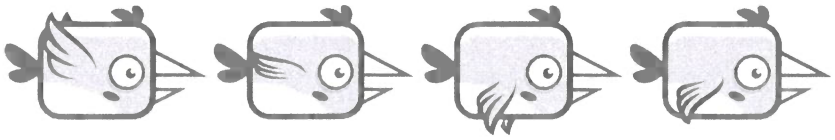
Nedan finns kod för en klass som hanterar fiender ihop med ett exempel som använder klassen. I funktionen `ClosestTo` är fyra datatyper ersatta mot en brädgård (#) och en stor bokstav. Observera att en bokstav kan finnas med flera gånger men representerar samma datatyp.

Studera koden och ange lämpliga datatyper som är utbytta. I ditt svar, ange vilken bokstav som hänger ihop med vilken datatyp.

```
class Run
{
   static void Main()
   {
      Enemies e = new Enemies();
      e.Add(new Vector2f(30, 11));
      e.Add(new Vector2f(33, 14));
      e.Add(new Vector2f(24, 8));
      Console.WriteLine(e.ClosestTo(new Vector2f(1, -3)));
   }
}


class Enemy
{
   public Vector2f position = new Vector2f(0, 0);
}


class Enemies
{
   private List<Enemy> instances;

   public Enemies()
   {
      instances = new List<Enemy>();
   }

   public void Add(Vector2f position)
   {
      Enemy e = new Enemy();
      e.position = position;
      instances.Add(e);
   }

   public #A ClosestTo(Vector2f position)
   {
      #B closest = 10000;
      for (#C i = 0; i < instances.Count; i++)
      {
         #D other = instances[i];
         float distance = other.position.DistanceTo(position);
         if (distance < closest)
            closest = distance;
      }

      return #A;
   }
}
```

Fråga 5 (5 poäng)

I pixelbaserade spel kan en bild som kallas *sprite sheet* användas för att animera en figur. Bilden i sig består av ett antal delbilder och ett exempel på en sådan bild finns nedan:



a) När spelet renderas visas bara en bildruta i taget, förklara hur det går till.

b) Förklara varför delbilderna samlas i en stor bild (istället för att varje bildruta har en egen bild).

c) Om vi vet bildens storlek och att delbilderna har en fast storlek, kan en delbild nås genom ett heltalsvärde (*index*). Förklara översiktligt hur översättnimgen från heltalsvärde till delbild görs.

Fråga 6 (4 poäng)

Skriv kod för en funktion som avgör om två vektorer är nära varandra. Funktionen tar in ett tröskelvärde som bestämmer vad som anses som nära. Nedan finns ett exempel där funktionen används, där ett sant värde kommer att skrivas ut.

Det finns ett förslag på lösning som du kan utgå från. Jämför vektorernas komponenter var för sig och avgör om differenserna är mindre än tröskelvärdet.

Du kan inte använda några inbyggda funktioner i vektorklassen.

Tips: Vektorernas komponenter kan nås via medlemmarna `X` och `Y`. Det går att räkna ut absolutvärde genom funktionen `MathF.Abs()`.

```
void Example()
{
    bool close = IsClose(new Vector2f(29.6f, -3), new Vector2f(29, -2), 1.1f));
    Console.WriteLine(close);
}

bool IsClose(Vector2f a, Vector2f b, float treshold)
{
    // TODO: Write function
}
```

Fråga 7 (3 poäng)

I schack gör spelare drag som innebär att en pjäs flyttas till en ruta. I en tänkt digital version av schack ska det finnas en funktion för att ångra drag d.v.s. återställa drag till innan de gjordes. Det ska vara möjligt att ångra drag ända till spelets början och det är viktigt att dragen kan ångras i den ordning som de spelades. För att lösa ångerfunktionaliteten lagrar koden dragen i en datastruktur. Det ska vara enkelt och effektivt att lägga till nya drag, att hämta ett drag och att ta bort det senaste draget. Det är två spelare i schack och dragen kan lagras i samma datastruktur eftersom spelarna alltid spelar varannan gång.

Välj en datastruktur som är lämplig att använda för att hantera dragen enligt ovan. Motivera också varför denna datastruktur är lämplig. Du behöver inte skriva någon kod. Ett svar utan motivering ger inga poäng.

Tips: Välj mellan datastrukturerna *List*, *HashSet*, *Dictionary*, *LinkedList*, *Queue*, *Stack*.

Fråga 8 (2 poäng)

Förklara vad nyckelordet *protected* innebär när det sätts på en klassmedlem. Ge också ett exempel på en kodsituation där detta nyckelord kan vara lämpligt att använda.