Uppgift 1 (2 poäng)

Man kan prata om tre typer av fel: kompileringsfel, *run-time* fel och/eller logiska fel. Kategorisera följande situationer (sätt **X**):

Situation	Kompileringsfel	*run-time* fel	logiskt fel
Multiplikation av två tal fast du tänkt addera dom			
Division med 0			
Felstavning av ord i utskrifter			
Du producerar felaktiga resultat			
Du skriver { när du skulle ha skrivit (			

Uppgift 2 (1 poäng)

Java sägs vara "starkt typat", vad menas med det?

Uppgift 3 (1 poäng)

Vad är fel med följande kodavsnitt? Använd block { } och ordna indenteringen så att utskrifterna blir rätt.

```
final int MAX = 24;
int total, summa;
...
if (total == MAX)
  if (total < summa)
    System.out.println("total är nu mindre än summa men lika med MAX");
else
  System.out.println("total är inte lika med MAX");
```

Uppgift 4 (1 poäng)

Hur kan överlagrade *(overloaded)* metoder skiljas från varandra?

Uppgift 5 (1 poäng)

Förklara skillnaden mellan överlagring (*overloading*) och omdefinition.

Uppgift 6 (3 poäng)

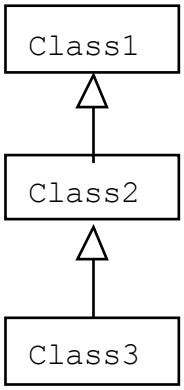
Beskriv den verkan olika modifierare har för olika strukturer (skriv e/t om kombinationen **e**j är **t**illåten):

	klass	**interface**	**metod**	**attribut**
`public`				
`protected`				
`private`				
`static`				
`final`				

Uppgift 7 (2 poäng)

Utgå från följande situation:

```
Class1 c1;
Class2 c2;
Class3 c3;
```



Ange om följande tilldelningar är korrekta eller ej. Motivera svaren!

	Ok	Ej Ok	Motivering
`c1 = c2;`			
`c2 = c1;`			
`c1 = c3;`			
`c3 = c2;`			

Uppgift 8 (3 poäng)

Designa klassen `Räknare` som realiserar en räknare där man kan summera ihop heltal en efter en och dessutom kan få reda på minimum, maximum och medelvärdet av de inmatade värden.

OBS! Själva inmatningen ska inte ske i `Räknare`. Anta att det finns andra klasser som ta hand om inmatningen som sedan använder sig av metoderna i `Räknare`.

Du behöver inte implementera metoderna.

Uppgift 9 (4 poäng)

Antag att följande metoder är deklarerade :

```
public void fixaElement (int[] minLista)
{
  minLista[2] = 77;
  minLista[4] = 88;
}

public void kopieraArray (int[] minLista, int[] minLista2)
{
  for (int index=0; index < minLista.length; index++)
    minLista[index] = minLista2[index];
}

public void flyttaReferens (int[] minLista, int[] minLista2)
{
  minLista = minLista2;
}

public int[] returneraReferens (int[] minLista2)
{
  minLista2[1] = 9876;
  return minLista2;
}

public void skrivLista (String utText, int[] minLista)
{
  System.out.println (utText);
  for (int index=0; index < minLista.length; index++)
    System.out.print (minLista[index] + "  ");
  System.out. print ln();
}
```

Utgå nu från följande fält:

```
int[] lista1 = {11, 22, 33, 44, 55};
int[] lista2 = {99, 99, 99, 99, 99};
```

Förklara vad som händer när följande kod exekveras (du ska starta med deklarationen ovan för varje deluppgift):

a)
```
fixaElement (lista1);
skrivLista ("Efter fixaElement:", lista1);
```

b)
```
kopieraArray (lista1, lista2);
skrivLista ("Efter kopieraArray:", lista1);
```

c)
```
flyttaReferens (lista1, lista2);
skrivLista ("Efter flyttaReferens:", lista1);
```

d)
```
lista1 = returneraReferens (lista2);
skrivLista ("Efter returneraReferens:", lista1);
```

Uppgift 10 (1 poäng)

```
int x = 1;
int y = 2;
int z = 3;

public void test()
{
  int y = 22;
  x = y;
  z = 33;
}

test();
System.out.println("x = " + x + "  y = " + y + " z = " + z);
```

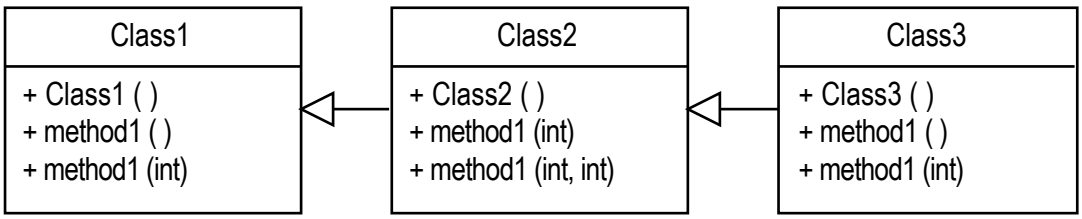
Vad skrivs ut?

Uppgift 11 (2 poäng)

Förklara syftet och innebörden med den speciella metoden `super(…)`.

Uppgift 12 (3 poäng)

Utgå från följande situation



och följande deklarationer:

```
Class1 c1 = new Class1();
Class2 c2;
Class3 c3;
```

Vilken *version* av `method1` anropas efter följande satser (skriv e/t om kombinationen **e**j är **t**illåten). Motivera dina svar!

	Klass	**Motivering**
`c1.method1();`		
`c1.method1(1);`		
`c1.method1(1, 1);`		
`c1 = c2;`		
`c1.method1();`		
`c1.method1(1);`		
`c1.method1(1, 1);`		
`c1 = c3;`		
`c1.method1();`		
`c1.method1(1);`		
`c1.method1(1, 1);`		

Uppgift 13 (2 poäng)

Beskriv skillnaden mellan en abstrakt klass och ett interface i Java.
Hur skiljer sig dessa från en "vanlig" klass?

Kladd och notiser