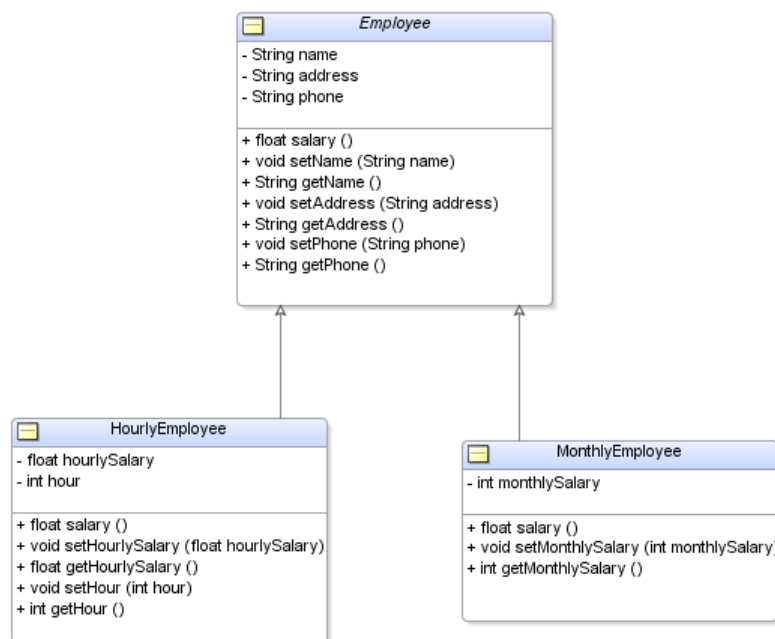




Laboration i JDeveloper



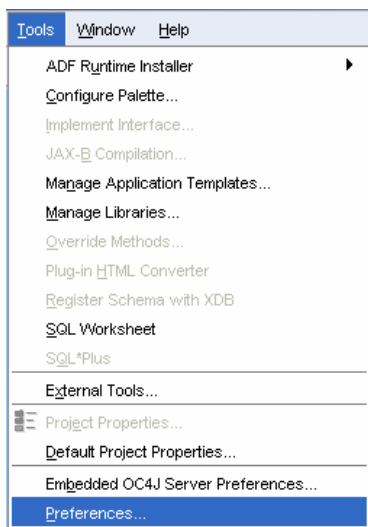
Starta JDeveloper och ändra inställningar

1. Dubbelklicka på symbolen för JDeveloper eller välj ifrån Startmenyn -> Programs -> JDeveloper -> JDeveloper. Se till att Oracle JDeveloper 10g startar. Det finns även tidigare versioner på datorerna.

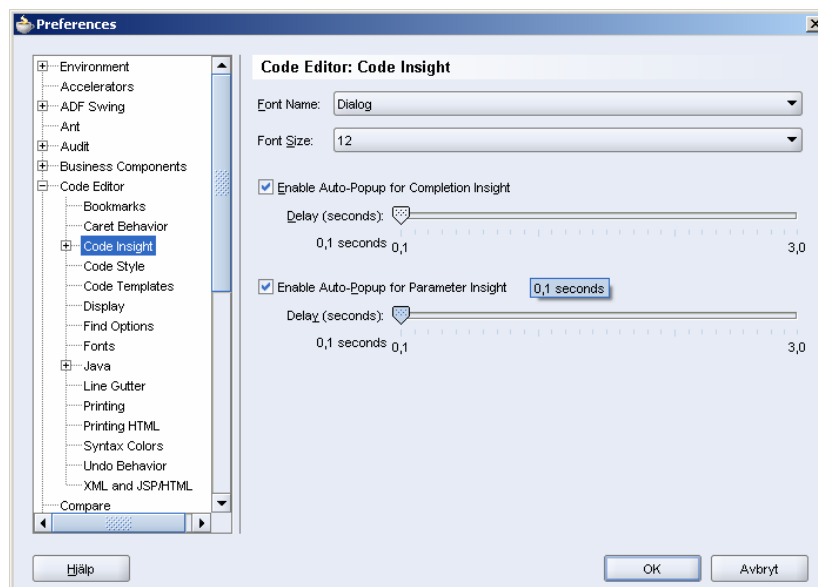


jdevw.exe

2. I Tools-menyn välj Preferences...



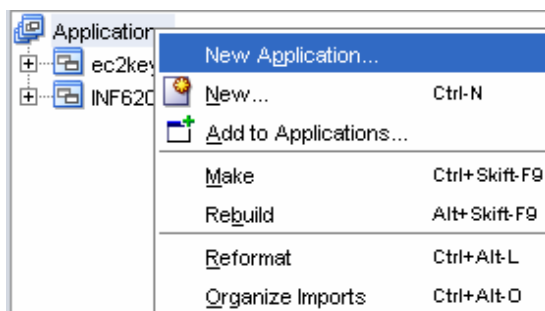
3. Välj Code Insight och minska reglagen för att få snabbare pop-up meny.



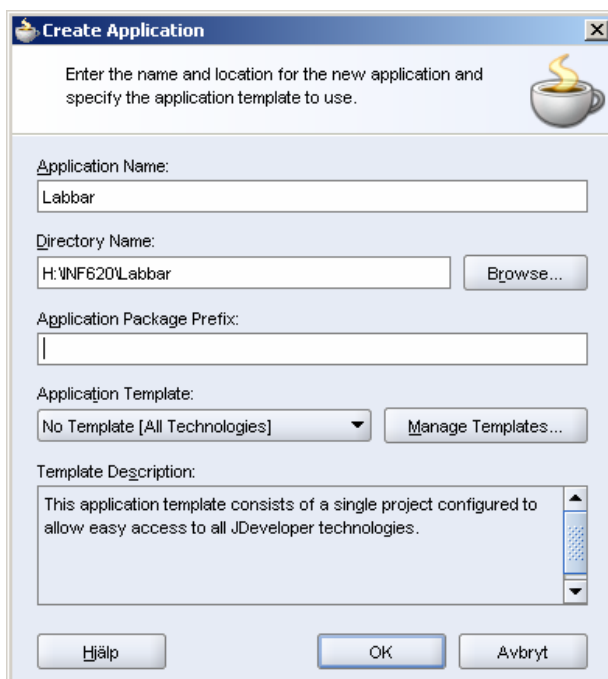
Skapa en ny *Application* och ett nytt *project*.

JDeveloper, använder *applications* för att hålla reda på de *project* (en samling av relaterade filer) som man använder när man utvecklar en applikation. Man kan ha flera *applications* öppna i *Navigatorn*. En *Application* kan innehålla många projekt. Ett projekt måste ingå i en *application*.

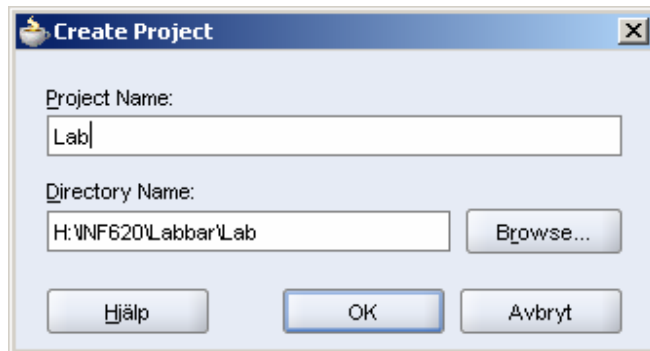
1. Klicka med högra mustangenten i *System-Navigator* fönstret på *Workspaces*, eller välj File -> New.
2. Välj *New Application*.



3. Ange katalog och namn, där du vill spara din applikation. Om du inte vill att din applikation skall ha ett package så ser du till att det fältet är tomt.



4. Ange sedan namn och katalog där du vill spara ditt projekt.

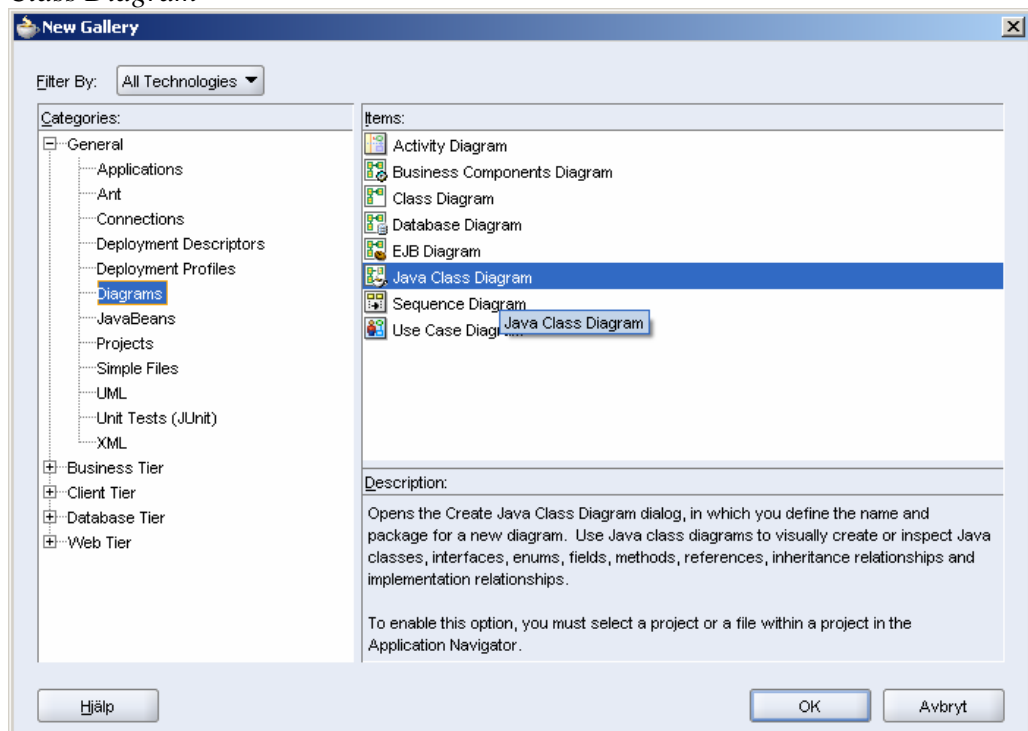


5. Nu skall du i *System-Navigator* fönstret se följande:

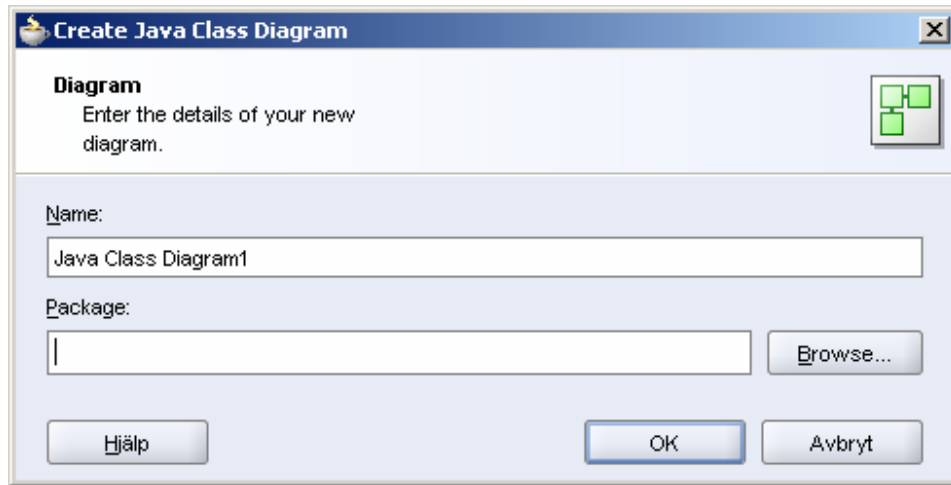


Skapa ett UML diagram.

1. Välj ur *System – Navigator* fönstret *New...* genom att klicka med högra musknappen på ditt projekt. Öppna *General* och markera *Diagrams*. Välj *Java Class Diagram*

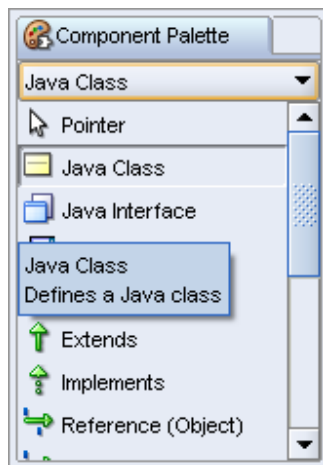


- Om man inte vill ha sina klasser i en *package*, töm detta fält.
Ange ett namn på ditt klassdiagram (default går bra det också)



Skapa en klass till ett diagram.

- Välj ur *Component Palette* Java Class. Se till att den är markerad.



- Klicka i *Class Diagram* fönstret där du vill ha klassen. Ange namnet *Employee* på klassen och avsluta med Enter.



3. Dubbelklicka på klassen för att få upp *Source* för klassen.

```
public class Employee {  
}
```

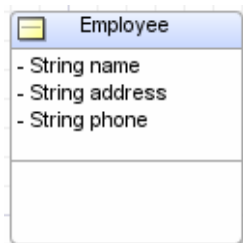
4. Se till att klassen verkligen heter **Employee** och att den är *public*, alla superklasser som inte ärver av någon klass ärver implicit av klassen **Object**.

Skapa attribut till en klass.

1. I *Source* fönstret till klassen, skriv in dina attribut i javakoden.

Attribut	Propeties	Value
name	Visibility	Private
	Type	String
address	Visibility	Private
	Type	String
phone	Visibility	Private
	Type	String

2. Modellen och *Source* är kopplade till varandra så modellen skall nu ha uppdaterats.



3. Filer som ej har sparats markeras i navigatorn med att de står i kursiv stil. Spara dina filer nu så att filnamnen står i vanlig rak stil.

Upprepa förfarandet för att skapa två nya klasser med attribut.

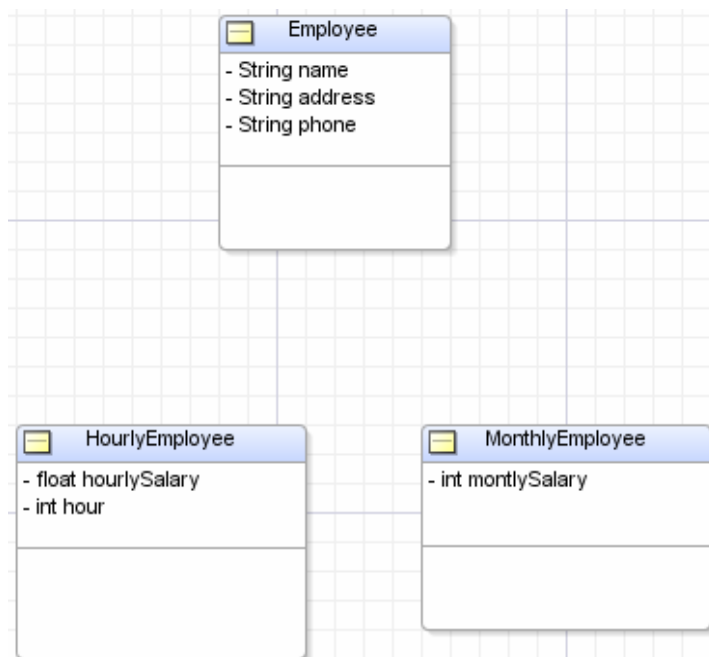
1. Öppna *Class Diagram* fönstret igen.
2. Klicka ut två nya klasser i diagrammet. (HourlyEmployee och MonthlyEmployee)
(Skapa två nya klasser på samma sätt som man skapade den första)
3. Skapa attributen i *Source* fönstret genom att dubbelklicka på respektive klass

HourlyEmployee:

Attribut	Propeties	Value
hourlySalary	Visibility	Private
	Type	float
hour	Visibility	Private
	Type	int

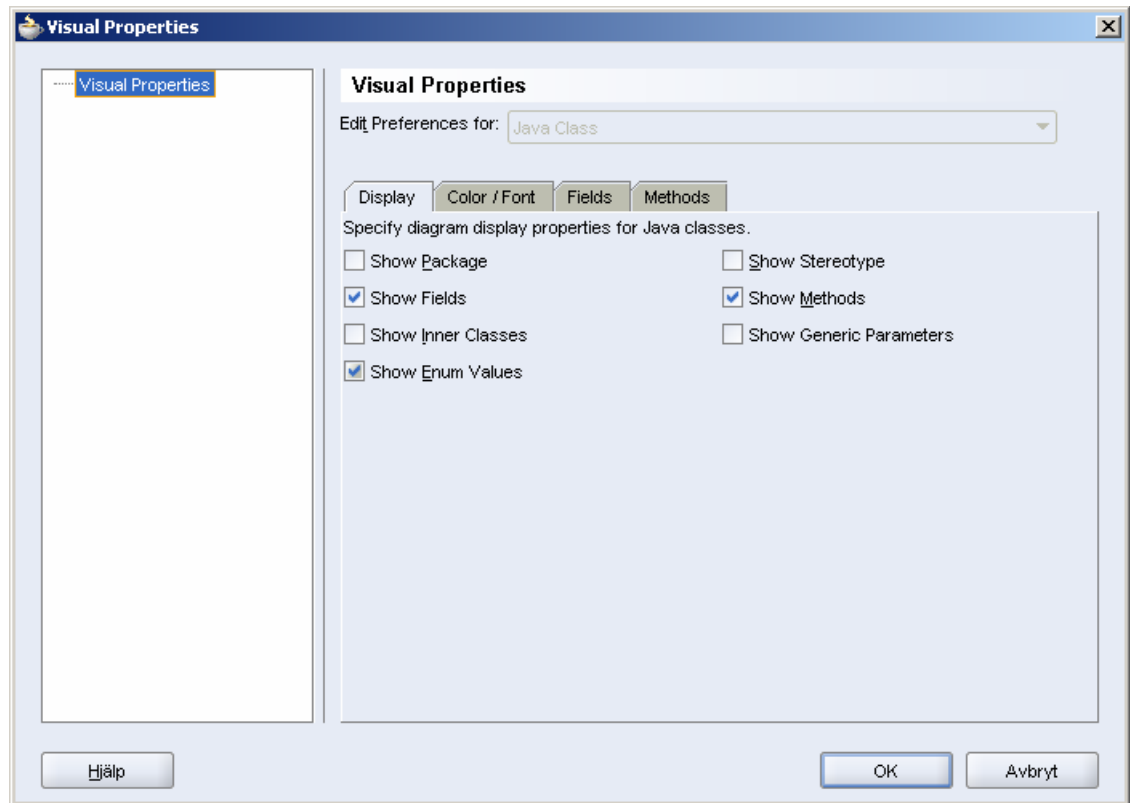
MonthlyEmployee:

Attribut	Propeties	Value
monthlySalary	Visibility	Private
	Type	int



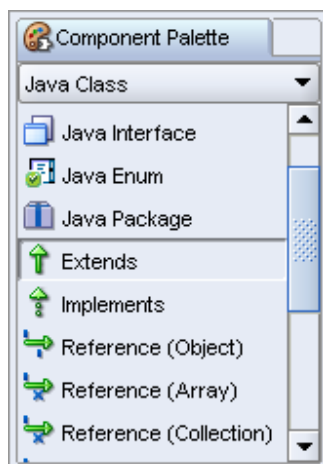
Ändra Visuella egenskaper

1. Markera en klass och högerklicka, i menyn välj **Visual Properties** (eller för hela projektet välj i **Tools** menyn **Preferences**, sedan **Diagram, Java Class**, och högst upp i dialogrutan **Java Class**. OBS! Inget som redan är genererat ändras).

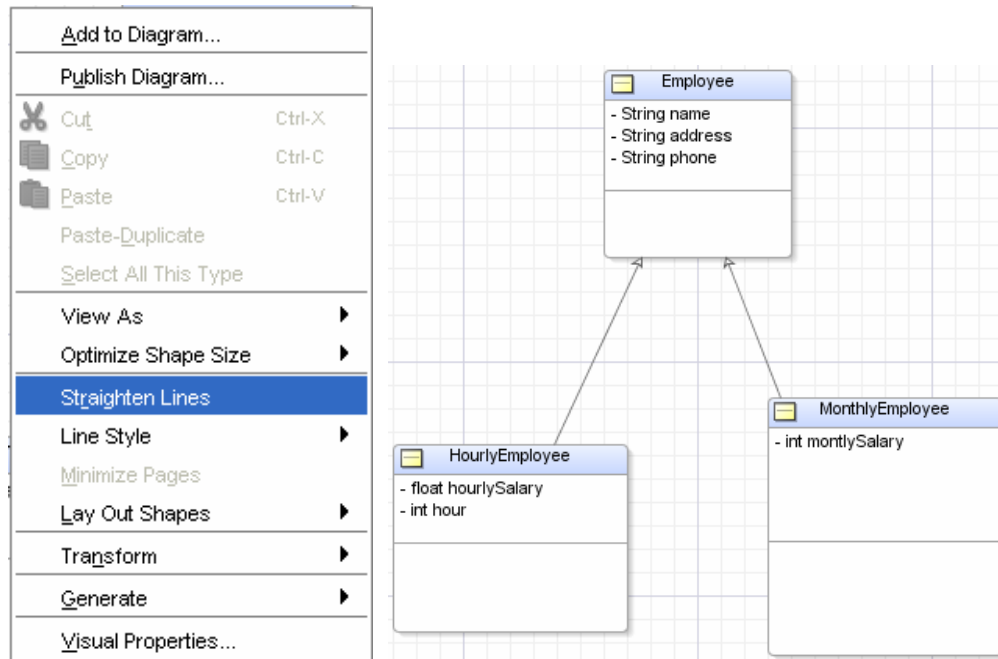


Skapa relation mellan klasser (Generalization)

1. Välj ut *Component Palette, Extends*



2. Klicka på subklassen och klicka sedan på superklassen.
3. Upprepa ovanstående punkter för den andra subklassen.
4. För att snygga till linjerna, högerklicka i diagrammet och välj *Straighten Lines*

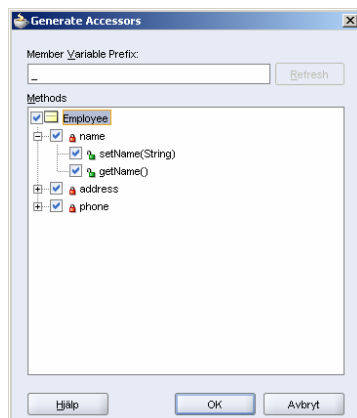


5. Titta på den genererade koden (.java filerna i *Navigatorn*).

Skapa åtkomstmetoder

JDeveloper kan generera åtkomstmetoder

1. Öppna *Suorce* för den klass du vill skapa åtkomstmetoder till.
2. Högerklicka i koden och välj *Generate Accessors*.



3. Klicka ok och se vad som genereras.

```
public String getName()
{
    return name;
}

public void setName(String newName)
{
    name = newName;
}
```

Modellen i Java Class Diagram uppdateras simultant.

4. Upprepa för de andra klasserna.
5. Gå till Diagrammet och titta på modellen. Om texten i rutorna inte får plats indikeras det med tre små prickar. Högerklicka i diagrammet eller på en specifik klass i diagrammet och välj ***Optimize Shape Size, Height and width.***
6. För att snygga till diagrammet, högerklicka igen och välj ***Lay out shapes, Hierarchical (Bottom to Top).***

Skapa metoder

1. Dubbel klicka på klassen Employee (för att få upp *Source* fönstret).
2. Skriv en ny publik abstrakt metod *salary(): float*.

```
public abstract float salary();
```

Editorn indikerar ett fel i klassdeklarationen. Detta beror på att en klass som har en eller flera abstrakta metoder måste deklarerats som abstrakt klass.

3. Ändra *Employee*-klassens huvud så klassen blir abstrakt.

```
public abstract class Employee
```

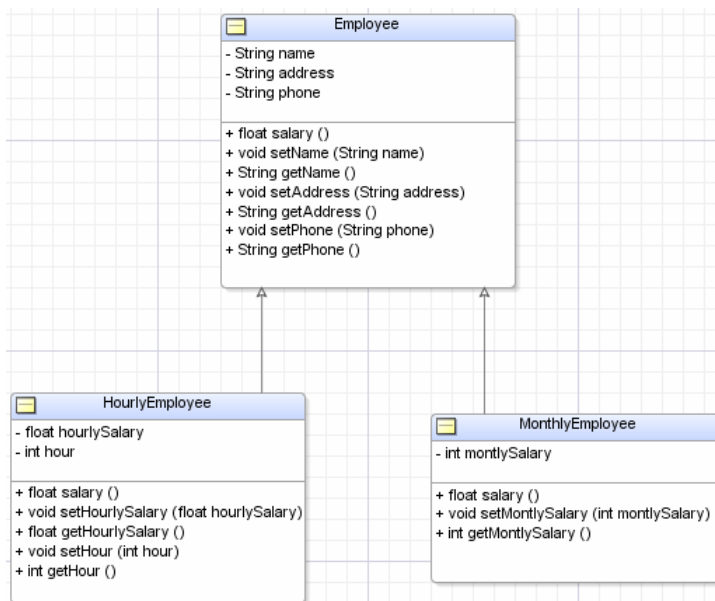
4. Skapa motsvarande metod för subklasserna. Dubbelklicka på HourlyEmployee (för att få upp *Source* fönstret för HourlyEmployee).

```
public float salary()
{
    return this.getHour()*this.getHourlySalary();
}
```



5. Upprepa förfarandet för klassen MonthlyEmployee d v s.
6. Dubbelklicka på MonthlyEmployee

```
public float salary()
{
    return this.getMonthlySalary();
}
```

7. Spara allt.

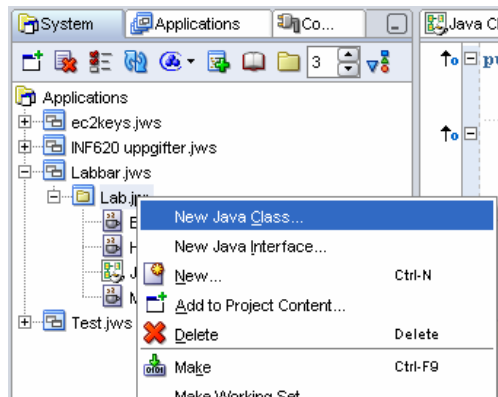


Kompilera.

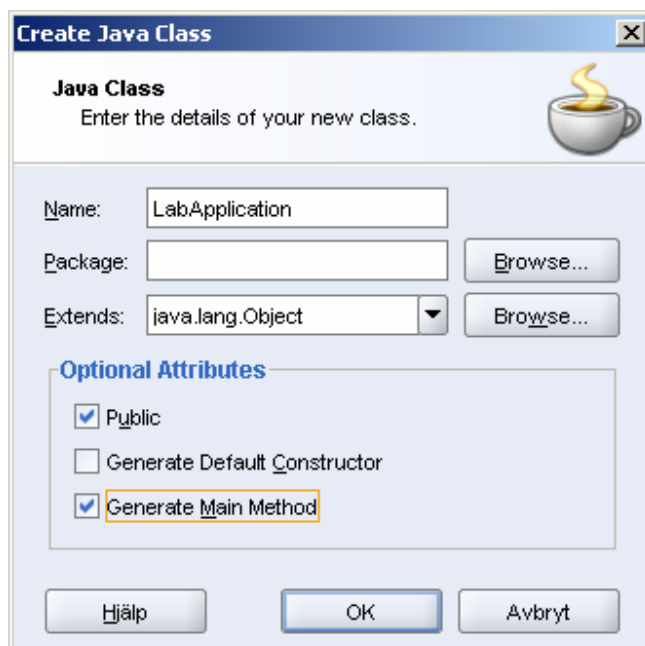
1. Kompilera ditt projekt genom att klicka på någon av följande symboler:   eller välj i **Project** menyn **Make/Rebuild**. **Make** kompilerar enbart om de klasser som ändrats sedan senaste kompileringen. **Rebuild** kompilerar alla klasser.
2. Ett tredje alternativ är att använda snabb tangenterna (ctrl-F9/alt-F9)

Skapa en applikationsklass


1. Markera i *System* fönstret ditt projekt och högerklicka.
2. Ur menyn välj *New Java Class*



3. Ange ett namn på din applikation under *Name*, (ta bort *Package*), klicka i *Generate Main Method* och klicka bort *Generate Default Constructor*.



4. Skriv in följande i *main(...)* metoden och kompilera:

 `public static void main(String[] args)`

```

{
    MonthlyEmployee F = new MonthlyEmployee();
    F.setName("Peter");
    F.setAddress("Norrviddinge");
    F.setPhone("0418-12345");
    F.setMonthlySalary(18000);
    System.out.println(F.getName()+" "+F.getAddress());
    System.out.println(F.salary());

    HourlyEmployee T = new HourlyEmployee();
    T.setName("Eva");
    T.setAddress("Malmö");
    T.setPhone("040-12121");
    T.setHour(100);
    T.setHourlySalary(70);
    System.out.println(T.getName()+" "+T.getAddress());
    System.out.println(T.salary());
}

```

Kör din applikation

1. Klicka på symbolen  eller välj Run ur menyn.
2. I message fönstret skall följande skrivas ut:

```

Peter Norrviddinge
18000.0
Eva Malmö
7000.0

```

Process exited with exit code 0.