Föreläsning 1

- Presentation av kursen
- Vad är programmering?
- Lite om java och utvecklingsmiljöer
- Aktivitetsdiagram
- Ett första program

Deitel: kapitel 1.1-1.14, 2.1-2.3

Programmering med Java – del I, DA101A

Kursansvarig: Kristina Allder
Tel: 040-6657630
Email: kristina.allder@mah.se

Kurssekreterare: Janet Stridh
Tel: 040-6657314
Email: janet.stridh@mah.se

Kursens hemsida:

- It´s learning: www.itslearning.com/elogin/default.aspx
- http://www.edu.mah.se/DA101A

Om kursen

Sidhänvisningar på kursen utgår från:

Java How to Program, Deitel & Deitel
Pearson Education (2008)
ISBN 9780136132479

Det kan gå bra att använda en annan bok om java.
Men sidhänvisningarna stämmer ej.

Varje vecka publiceras följande på **it´s learning**:

- Föreläsningsunderlag
- Kommentarer till föreläsningarna
- Laborationsmaterial

Under kursens gång ska du lösa 5 inlämningsuppgifter.

Kursen avslutas med skriftlig tentamen. Tentamen är betygsgrundande.

Vad är programmering?

Det finns många svar på den frågan, beroende på vilken infallsvinkel man har på ämnet och hur man uppfattar frågan. Här nedan är några förslag på svar. Hur ser din uppfattning ut? Sammanfaller den med någon av de nedanstående alternativen?

- Att ge instruktioner till en maskin så att den utför det man vill
- Del av att utveckla programvara
- Kreativ problemlösande verksamhet

Att göra ett program

- **Uppgiftsformulering**, vad är det för uppgift som ska lösas? Formulera uppgiften i termer av vad en dator kan utföra. Avgränsa problemet, vad är en del av uppgiften? Vad ingår inte?
- **Algoritmkonstruktion**, vilka algoritmer är de mest lämpliga för detta problem? Konstruera strukturen på programmet och skriv ner så kallad pseudokod. Detta är kreativ problemlösning.
- **Kodning**, översätt pseudokoden till ett programmeringsspråk t.ex. Java
- **Dokumentation**, beskriva din lösning både i löpande text, med hjälp av UML och som kommentarer i programmet (t.ex. javadoc).
- **Verifikation**, är programmet byggt på ett bra sätt så att det löser uppgiften utan att fel uppstår och det är lätt att underhålla.
- **Validering**, är användaren nöjd med hur programmet fungerar.
- **Underhåll**, åtgärda buggar, förbättra och lägg till funktionalitet.

Java lite historia

- Java är ett programspråk som utvecklats av *Sun Microsystem* under ledning av *James Gosling*. Det blev allmänt tillgängligt 1995. Från början var tanken med Java att det skulle användas i olika typer av elektronik som brödrostar eller diskmaskiner.
- 1995: Från början var Java mest känt som ett slags programspråk som man använder på Internet för att skapa häftiga effekter på webbsidor. Javaprogrammet kan t.ex. generera ljud och rörliga bilder eller låta användaren kommunicera med programmet med hjälp av mus och tangentbord.
- Java är ett fullfjädrat programspråk. Med Java kan man, liksom t.ex. C++, skapa fullständiga applikationsprogram.

Egenskaper hos Java

- **Java är plattformsoberoende**.
  Med *plattform* menas ett visst operativsystem som kör på en viss typ av dator. Windows XP på en PC är t. ex. en plattform. Linux på en PC är en annan plattform.
- **Java innehåller verktyg för att generera grafiska användargränssnitt**.
  Man kan alltså med hjälp av Java skriva grafiska program, dvs sådana program som använder fönster, menyer, knappar etc, för att kommunicera med användaren.
- **Java är objektorienterat.**
  Java bygger helt på de objektorienterade principen för att konstruera program.
- **Java gör det möjligt att skriva parallella program.**
  Java stödjer nämligen s.k. multitrådar. Detta innebär att javaprogrammet kan beskriva flera aktiviteter som pågår samtidigt.
- **Java kan användas på Internet.**

Några fördelar/nackdelar med Java

- Att köra ett javaprogram går lite långsammare än att köra motsvarande program i t.ex. C++. Detta för att javaprogram (t.ex Welcome.java) ska kunna köras på olika plattformar utan att modifieras.
- Java har bibiliotek med många standardklasser som man bör sätta sig in i och använda. De finns i Javas **API** (**A**pplication **P**rogram **I**nterface) och vi använder **JSE** (**J**ava **S**tandard **E**dition). Javas API innehåller dock oerhört många klasser så det är endast möjligt att använda en mindre del av dem utan att ta del av dokumentation.

Java (IDE)

Ett modernt integrerat programutvecklingssystem, ett s.k. IDE (**I**ntegrated **D**evelopment **E**nvironment) brukar erbjuda programmeraren en miljö där det finns alla hjälpmedel man behöver för att utveckla program. Det finns sådana system även för Java, t.ex.

- **JDK1.0, JDK1.0.2, JDK1.1, JDK1.2,JDK1.3, JDK1.4.2, JDK 1.5, JDK 6**
- **Eclipse** (www.eclipse.org)
- **NetBeans 6.5 från Sun** (www.netBeans.org)
- **Turbo JBuilder 2008** (http://www.borland.com/us/products/ide.html)

Java (JDK)

- Sun har utvecklat det så kallade **JDK** (**J**ava **D**evelopment **K**it) som är ett programpaket som innehåller de olika program och klassbibliotek som man behöver för att utveckla och köra javaprogram. JDK är allmänt tillgängligt på internet.
- I kursen kommer JDK 1.6 och Eclipse att användas.

Programmeringspråk

Ett programmeringsspråk är ett antal regler (syntax) och dess betydelser (semantik) i vilket en programmerare kan uttrycka sig och ge instruktioner till en dator.

Programmerare skriver instruktioner (text) i programmeringsspråket. Instruktionerna översätts av en kompilator till kod som kan exekveras på en dator.

Det finns ett stort antal olika programmeringsspråk som man kan använda. Ett par vanliga är:

- C++
- C#
- Java
- Visual Basic

Aktivitetsdiagram

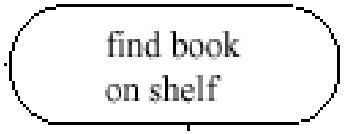
Ett sätt att beskriva en algoritm eller ett flöde i ett program är med hjälp av ett aktivitetsdiagram. Aktivitetsdiagrammet är ett av nio diagram som ingår i UML-standaren. UML är ett standardiserat sätt att beskriva datorprogram/system som användas både vid konstruktion och dokumentering.

I kursen kommer aktivitetsdiagram främst att användas för att beskriva algoritmer.

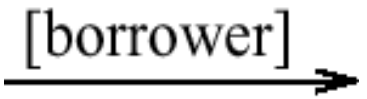
Aktivitetsdiagram (Activity diagram)

De olika symbolerna som ingår i aktivitetsdiagram:

- Activity (beskriver en aktivitet)

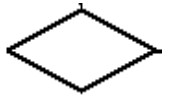


- Transition with guard (en övergång från en aktivitet till en annan med ett vilkor som måste vara uppfyllt för att övergången får ske.)



- Synchronization bar (fork, join) (för att markera att detta sker samtidigt)

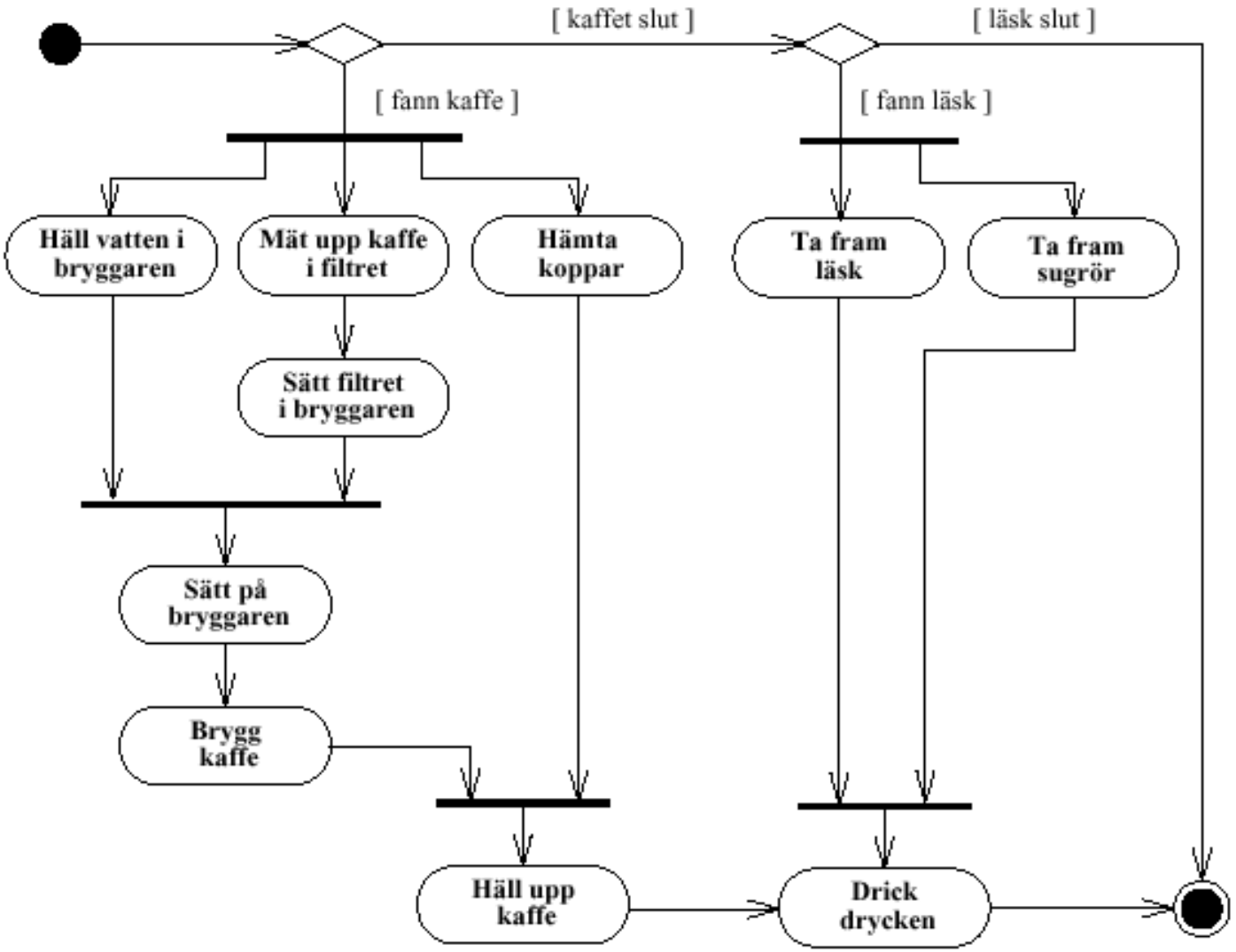
- Decision or merge (ett vägskäl eller en ihopslagning för att kunna välja olika vägar beroende på ett val)



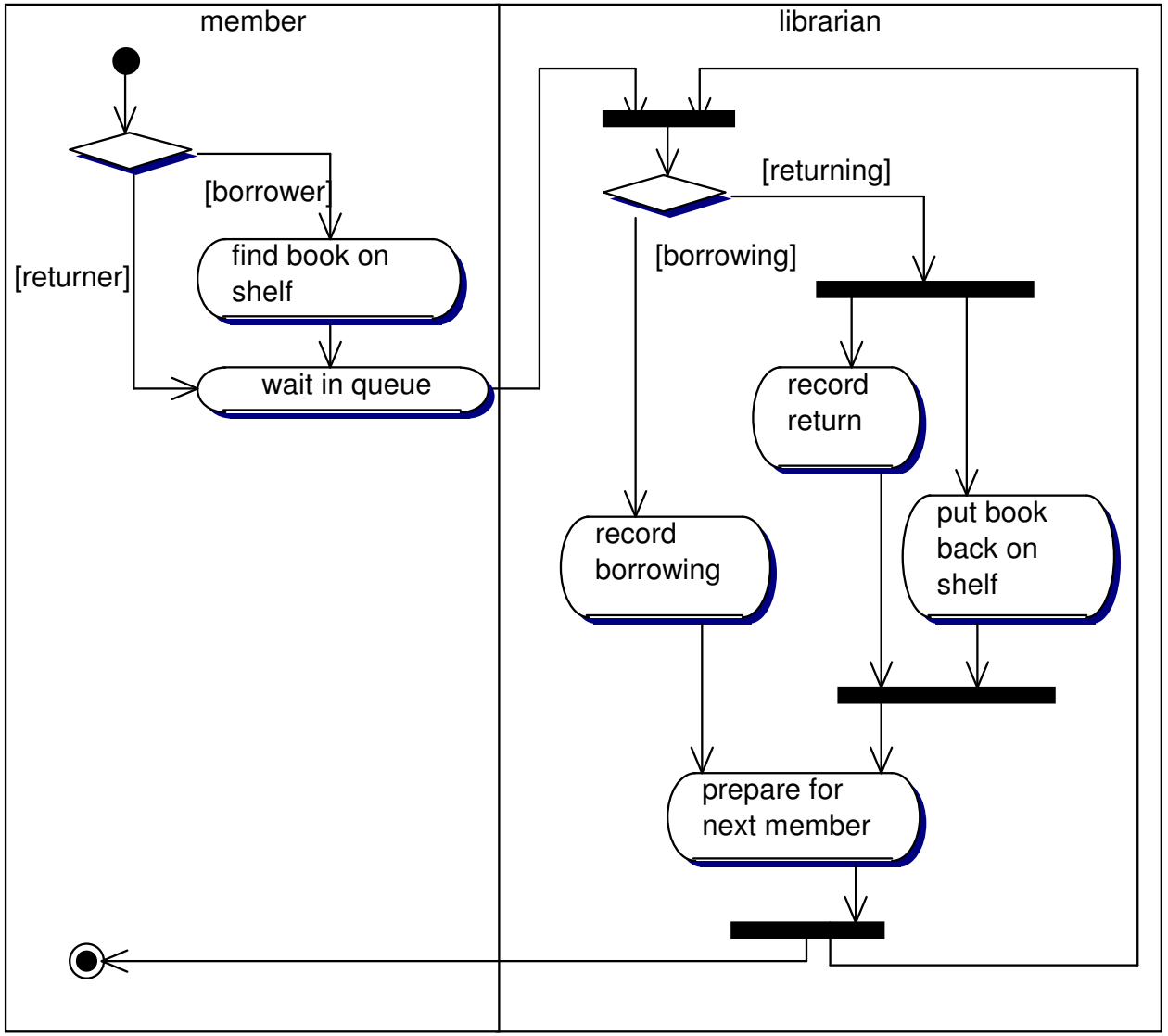
- Start marker (aktiviteten börjar)

- Stop marker (ett av kanske flera möjliga avslutningar)

Aktiviteten ”att fika”



Aktiviteten att låna böcker



Hur programmerar man?

1) Man skriver in programkoden (källkoden) med hjälp av en texteditor
2) Man sparar koden (som enbart består av text) i en fil. När man arbetar med Java ska filen alltid ha suffixet java, dvs. filnamn.java
3) Man kompilerar (översätter) filen till ett format som datorn kan förstå. När man arbetar med Java översätts källkoden till ett format som kallas bytekod (filnamn.class). Bytekoden interpreteras (tolkas) sedan av en virtuell maskin i datorn (alternativt kompileras till maskinkod som körs på datorn).
4) Man testar programmet för att se om det fungerar som man hade tänkt sig. Om det inte fungerar som man hade tänkt sig letar man upp felet, ändrar i källkoden och testar igen.

Olika typer av fel

- Kompileringsfel
  Dessa fel uppstår redan när man försöker att kompilera programmet, dvs. Javakompilatorn skriver ut ett felmeddelande. Ett vanligt exempel är att man han stavat fel på något ställe i källkoden eller glömt ett semikolon. Här kollar kompilatorn om syntaxen är korrekt.
- Exekveringsfel
  Dessa fel uppstår när man exekverar (kör) programmet, det vill säga man får ett felmeddelande från Javainterpretatorn. Ett vanligt exempel är att man försöker dividera ett tal med noll (någonting delat med noll ger ett odefinierat resultat)
- Logiska fel
  Dessa fel visar sig genom att programmet inte gör det man hade tänkt, det vill säga programmet kör utan felmeddelanden men resultatet blir inte det avsedda. Den här typen av fel är de som är svårast att hitta.

Java Applets och Java Applikationer

Det finns två typer av Javaprogram:

- –Java Applets
- –Java Applikationer

- De Javaprogram som man hittar på olika hemsidor på Internet är nästan alltid Applets. Bytekoden till Applets-program byggs in i HTML-dokument och körs med hjälp av Webbläsaren (Netscape eller MS Explorer), dvs. Webbläsaren sköter interpreteringen av bytekoden.
- Java Applikationer fungerar som mer traditionella datorprogram, och de körs med hjälp av den vanliga interpretatorn (som heter java.exe).

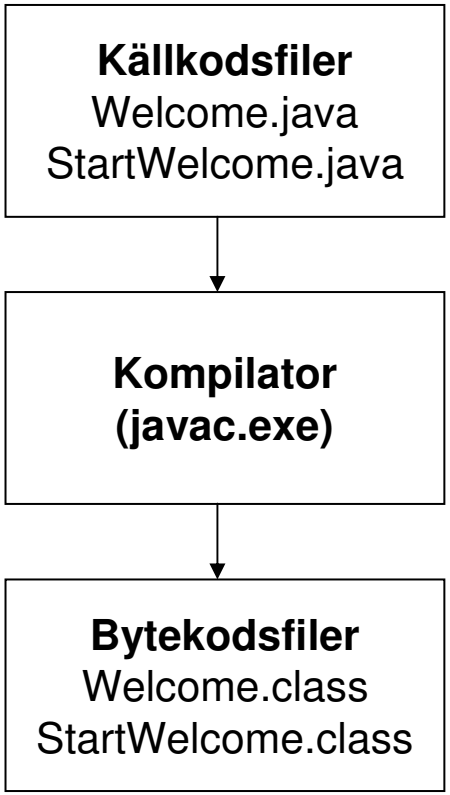
Ett enkelt program – skriva källkoden

```
//This program prints  Welcome to Java!
public class Welcome {
  public void sayHello() {
    System.out.println("Welcome to Java!");
  }
}

public class StartWelcome {
  public static void main(String[] args) {
    Welcome application = new Welcome();
    application.sayHello();
  }
}

/*
De program som du skriver på kursen kommer som regel att
använda minst två klasser:
• Klass med main-metod i vilken exekveringen startar
• Klass med metoder som som utgör de väsentliga delarna
  av programmet
*/
```

Ett enkelt program – kompilera källkoden

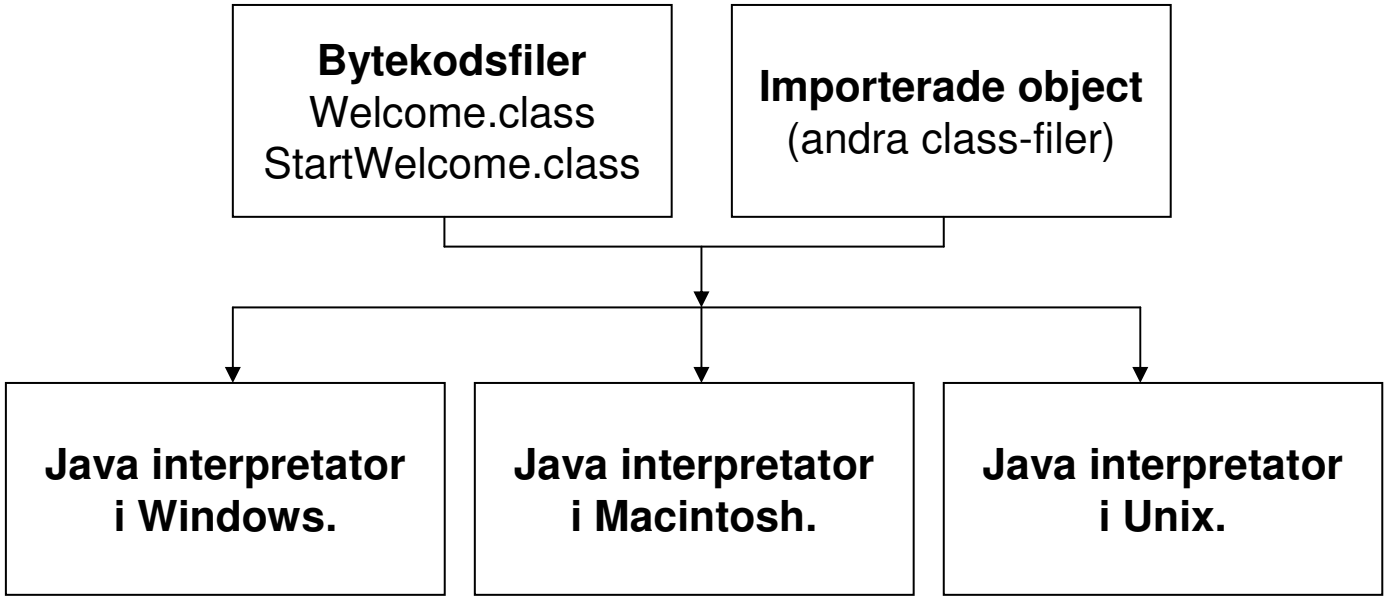


Källkodsfilerna läses av programmet javac.exe

javac.exe översätter källkodsfilerna till bytekodsfiler

Bytekodsfilerna används när programmet ska exekveras (köras)

Ett enkelt program – exekvera bytekoden



Bytekodsfilerna och filer som importeras tolkas av Java-interpretatorn (JVM – Java Virtual Machine, t.ex. java.exe) och ger ett körresultat.

Ungefär samma körresultat uppnås på olika plattformar.

Att köra ett program i Eclipse / NetBeans

Laboration 1 beskriver hur du ska arbeta med Eclipse / NetBeans 6.5.

Filer du behöver är:

DA101AL1VT09.pdf + Eclipse.pdf / NetBeans6_1.pdf + eventuellt JSE.pdf

Filerna finner du i L1.zip.