

Grundläggande programmering

Idag

- Kursinformation
- Kursinnehåll
- Java i allmänhet
Varför?
- Exempel ett komplicerat Javaprogram från slutet av kursen
Svårt
- Programspråkskonstruktioner i Java
Från början - enkelt

Grundläggande programmering

Kursen

Anders Berglund

föreläsningar, lektioner, kursansvar

Jesper Bengtsson

föreläsningar, lektioner, labbar, handledning

Bra resurser

Bennie Lundmark

SI - extraundervisning, schemalagd, frivillig

DatorQ

handhavandehjälp

WWW

För labbar, meddelanden etc.

<http://www.it.uu.se/edu/course/homepage/grundprog/vt05>

Undervisningsformer

- **Föreläsningar**
 - Översiktskaraktär.
 - Reservföreläsningar är reserv (!)
 - Preliminär föreläsningsplan. Ändringar via webben
 - Gästföreläsning.
 - Tips: Skriv inte av OH-bilder
 - Läs Skansholm som uppslags/problemlösningbok

Undervisningsformer

- **Lektioner/Övningar** Halvklass.
 - Förberedelseuppgifter som ger bonus
 - Studenter vid tavlan
- **Seminarier**
 - Ett med kursen vetenskapsteori (Andra delen av maj, ännu ej schemalagt)
 - Ett för programmeringsuppgiften
- **Laborationer** Tredjedelsklass
 - Redovisningsformerna varierar
 - Lämna in i tid
 - Redovisas/Genomförs vid våra datorer
 - Tips: Kolla ditt konto idag
- **Programmeringsuppgift**
 - Lite större, lite svårare.
 - Seminarium, som liknar en disputation
- **Handledning**
 - Respektera Jespers mottagningstider.
 - Tips: DatorQ för handhavande frågor
- **Supplemental Instruction (SI)**
 - Komplement till föreläsningar/lektioner
 - Studentledd aktivitet

Laboration 1

- Knappningsövning. Rätt tillfälle repetera Unix etc.
- Busenkel.
- Det är handhavandet, inte programkoden eller vad koden ”betyder” som har betydelse denna gång
- Alla frågor är bra frågor.
- Obligatorisk närvaro, inget att redovisa.
- Förberedelse: Kolla att ditt lösenord fungerar. Logga in på ditt konto DatorQ (1203, kl. 9 - 17)
- Samling i en lektionssal

Examination

- **Obligatoriskt**
 - laborationer
 - programmeringsuppgift
 - seminarier
 - tentamen
- **Graderat betyg**
 - beror av tentamensresultatet (+ bonus)
- **Bonus**
 - för *förberedelser* av lektionsuppgifter
 - 2% av tentamens totala poäng per lektionstillfälle
 - närvaro vid lektionen.

Kursinnehåll

Förutsättningar

- **Programmering, objekt-orienterad**
- **Problemlösning med datorer**
- **Java ett redskap, inte en kurs i Java**
- **Nybörjarkurs**

leder till

- **Få Javakonstruktioner**
- **Ganska mycket grundläggande begrepp**
- **Mycket strukturering av problem**

Kursinnehåll

Från föreläsningsplanen

- **Vad är programmering?**
Programkonstruktion, algoritmbegreppet.
- **Introduktion till objektorienterad programmering och Java.**
- **Klasser, objekt, metoder. Variabler, tilldelning, enkel inläsning/utskrift. Metoder, parametrar, styrande satser**
- **Enkel objektorientering.**
- **Avancerad objektorientering, arv, relationer**
- **Beräkningsmodeller**
- **Användning av färdiga klasser, exempel**
- **Enkel grafik**
- **Introduktion till projekten. Större program, avancerade programspråkskonstruktionerna**
- **Dataetik**

Spelet tjugoett

Jämfört med tidigare år ...

- Samma innehåll, samma grundstruktur, liknande labbar, liknande lektioner, samma bok.
- Arbetsfördelning som ifjol: Mest föreläsningar, lektioner, labbar i början, mest projekt i slutet.
- Förändringar efter kursutvärderingar:
 - Lektioner och seminarier i vetenskapshistoria på olika dagar
 - Mer bonus för lektionsuppgifter
 - Mer föreläsningstid för stora programexempel
 - ”Mittredovisning” för projektet
 - Viss storleksgradering av projekten
- Eget arbete/SI verksamheten *ännu mer* i fokus

- Stort, svårt exempel från slutet av kursen.
- Kursboken, kap 10.10
- Visar vad vi kan vid kursens slut - inrymmer många av kursens begrepp.
- Det finns *många* lösningsstrategier - inget "rätt svar". Vald lösning finns i kursboken.
- Fokusera på idéer, inte på programspråkskonstruktioner

Spelet tjuogoett

Du spelar mot banken

Välkommen till tjuogoett
Du fick Ruter Ess och har 14 poäng
Ett kort till? j
Du fick Hjärter 6 och har 20 poäng
Ett kort till? n
Datorn fick Klöver Ess
Datorn fick Klöver 2
Datorn fick Klöver 10
Datorn fick Spader 9
Datorn har 22 poäng
Du vann!
Nytt parti? j
Du fick Hjärter 8 och har 8 poäng
Ett kort till? j
Du fick Spader Kung och har 21 poäng
Du vann!
Nytt parti? j
Du fick Spader 9 och har 9 poäng
Ett kort till? j
Du fick Spader 5 och har 14 poäng
Ett kort till? j
Du fick Klöver 9 och har 23 poäng
Du förlorade!
Nytt parti? n

"Vilken Java" finns i programmet?

- Någon form av minnesfunktion
- Någon form av repetition
- Någon form av kommunikation mellan objekt
- Någon form av felhantering
- Lite annat

```

public class Tjugoett {
    Kortbunt lek = new Kortbunt();
    Manniska du = new Manniska(lek);
    Dator jag = new Dator(lek, du);

```

*Här ska metoden Tjugoett() vara,
som startar spelet och avgör vem som vinner*

```

    public static void main(String[] arg)
    {
        Tjugoett tj = new Tjugoett();
    }
}

```

```

Tjugoett() {
    System.out.println("Välkommen
        till tjugoett");
    boolean nyttPartiÖnskas = true;
    while (nyttPartiÖnskas) {
        lek.nyKortlek();
        lek.blanda();
        du.spela();
        if (du.poäng() > 21)
            System.out.println("Du
                förlorade!");
        else if (du.poäng() == 21)
            System.out.println("Du vann!");
        else { // datorn måste spela
            jag.spela();
            if (jag.poäng() <= 21 &&
                jag.poäng() >= du.poäng())
                System.out.println("Du
                    förlorade!");
            else
                System.out.println("Du
                    vann!");
        }
        System.out.print("Nytt parti? ");
        nyttPartiÖnskas = du.svararJa();
    }
}

```

```

public abstract class Spelare {
    protected Kortbunt leken;

    protected Kortbunt hand =
        new Kortbunt();
    protected int p;

    public Spelare(Kortbunt k) {
        leken = k;
    }

    public abstract void spela();

    public void nyttSpel() {
        hand.slängKorten();
        p = 0;
    }

    public int poäng() {
        return p;
    }
}

```

```

import java.io.*;

public class Manniska extends Spelare {
    BufferedReader
        myIn = new BufferedReader
            (new InputStreamReader(System.in));

    public Manniska(Kortbunt k) {
        super(k);
    }

    public void spela() {
        boolean nyttKortÖnskas = true;
        nyttSpel();
        while (p < 21 && nyttKortÖnskas) {
            System.out.println("Du fick " +
                nyttKort() +
                " och har " + p + " poäng");
            if (p < 21) {
                System.out.print("Et till? ");
                nyttKortÖnskas = svararJa();
            }
        }
    }
}

```



```

public boolean svararJa() {
    System.out.flush();

    String s = "";
    try {
        s = myIn.readLine();
    }
    catch (IOException e) {}
    return s.equals("") ||
           s.equals("j") ||
           s.equals("ja");
}
}

```

```

public class Dator extends Spelare {
    private Spelare motspelare;

    public Dator(Kortbunt k, Spelare mot){
        super(k);
        motspelare = mot;
    }

    public void spela() {
        nyttSpel();
        while (p < 21 && p <
               motspelare.poäng()) {
            System.out.println("Datorn fick "
                               + nyttKort());
        }
        System.out.println("Datorn har " + p
                           + " poäng");
    }
}

```

```

public Kort nyttKort() {
    Kort k = leken.geÖversta();
    hand.läggÖverst(k);
    int antalEss = 0;
    p = 0;
    for (int i=1;
        i<=hand.antalKort(); i++) {
        int v = hand.tittaPå(i).valör();
        if (v == 1) { // ett Ess
            p += 14;
            antalEss++;
        }
        else
            p += v;
    }
    for (int j=1; j<=antalEss && p>21;
        j++)
        p -= 13;
    return k;
}
}

```

```

public class Kortbunt {
    private Kort[] bunten = new Kort[52];
    private int antal = 0;

    public int antalKort() {
        return antal;
    }

    public void slängKorten() {
        antal = 0;
    }

    public Kort tittaPå(int nr) {
        return bunten[antal-nr];
    }

    public void läggÖverst(Kort k) {
        bunten[antal++] = k;
    }

    public Kort geÖversta() {
        return bunten[--antal];
    }

    public void nyKortlek() {
        slängKorten();
        for (int f=1; f<=4; f++)
            for (int v=1; v<=13; v++)
                läggÖverst(new Kort(f,v));
    }
}

```

Vad såg vi?

Programspråkskonstruktioner för att

- Spara värden
- Repetera
- Definiera objekt
- Beskriva relationer mellan objekt
- Mata in eller ut data

Objektorientering

- Sätt att modellera världen med objekt
- Klasser beskriver typer av objekt
- Objekten interagerar
- Objekten har egenskaper/tillstånd och kan handla

```
public void blanda() {  
    for (int i=1; i<1000; i++) {  
        int n1 = (int) (Math.random() *  
            antal);  
        int n2 = (int) (Math.random() *  
            antal);  
        Kort temp = bunten[n1];  
        bunten[n1] = bunten[n2];  
        bunten[n2] = temp;  
    }  
}
```

Varför lär man sig detta?

- Detta \neq "Bara" ren programmering.
- Förstå datorer/datoranvändning är en del av ingenjörskompetens.
- Dator/programmeringskunskap med grundbegrepp är nyttigt i sig och ganska stabilt. Tillämpningsprogram förändras blixtnabbt
- Ger en problemlösningstrategi.
- Speciellt för STS: Kunna kommunicera med olika specialister.
- Intressant för en generalist.

Hur lär man sig detta?

- *Programmera/Lös problem själv*
- Fråga, diskutera, samarbeta
- *Programmera/Lös problem själv*
- Utnyttja SI
- *Programmera/Lös problem själv*
- Jobba mycket i början. Det blir snabb svårare
- *Programmera/Lös problem själv*
- Använd Skansholm och WWW som uppslags/problemlösning böcker
- *Programmera/Lös problem själv*
- Använd föreläsningar för överblick, stora linjer
- *Programmera/Lös problem själv*