

Föreläsning 3

Tobias Wrigstad

Resten av kursen...



Så här långt borde du ha...

1. Gått på två föreläsningar
2. Gjort klart 4/6 labbar
3. Ha tittat lite på kursens webbsida
4. Ha skaffat ett GitHub-konto och registrerat det hos oss
5. Ha loggat in på Piazza minst en gång
6. Ha loggat in på Rocket.Chat minst en gång

**Från och med nästa vecka: Labbar på
IOOPM avser tid att redovisa och få
hjälp med programmering.**

**Den mesta programmeringen sker
utanför schemalagd tid!**



Din uppgift

Att lära dig så mycket som möjligt om imperativ- och objektorienterad programmering, verktyg, programmeringsmetodik och process

Vår uppgift

Att hjälpa dig i din inlärningsprocess.

Att se till att det leder till att du blir godkänd på kursen.



IOPM 2017

Tobias Wrigstad

Kursansvarig

Elias Castegren

Huvudassistent



Imperativ och ObjektOrienterad ProgrammeringsMetodik



Imperativ och ObjektOrienterad ProgrammeringsMetodik

Del 1 & 2



Imperativ och ObjektOrienterad ProgrammeringsMetodik

Del 2



Imperativ och ObjektOrienterad ProgrammeringsMetodik

Del 1, 2 & 3



Efter godkänd kurs ska studenten kunna [1/2]

- förklara hur program **exekverar** och beskriva hur program **lagrar och hanterar information**.
- förklara och exemplifiera **kvalitativa** och **kvantitativa aspekter** av ett programs **design** och **implementation**.
- förklara skillnaden mellan **manuell och automatisk minneshantering** samt grundläggande relaterade begrepp (som stack, heap, statisk minnesarea) och använda **verktyg** för felsökning av minnesfel.
- förklara hur **större programuppgifter** kan lösas & resonera om olika lösningsalternativ.
- redogöra för hur **enkla parallellicerbara problem kan lösas** effektivt med relevanta hjälpmittel.
- **designa, koda, granska, testa, felsöka och dokumentera** egna program, med hjälp av lämpliga **verktyg**.
- **läsa, förstå och modifiera** icke-trivial **kod** som studenten själv inte har skrivit samt **integrera nyskriven kod med existerande**.



Efter godkänd kurs ska studenten kunna [2/2]

- beskriva – **oberoende av programkoden** – den uppgift ett program skall lösa och de förutsättningar som krävs för att det skall kunna arbeta. Motivera varför programmet under dessa förutsättningar är korrekt, **samt specificera och konstruera testfall och köra tester för att verifiera detta.**
- skriva lämplig **dokumentation för programmering** och **testhantering**.
- tillämpa specifika **utsnitt av kända utvecklingsprocesser** och -metodiker (ex. **agil utveckling, parprogrammering** och testdriven utveckling).
- beskriva olika former av **testning** och deras vikt i olika skeden av **utvecklingsprocessen**.
- bidra till ett **konstruktivt samarbete i programmeringsprojekt**.
- **presentera** och **diskutera** kursens innehåll **muntligt** och **skriftligt** med för utbildningsnivån lämplig färdighet.



Mål

Unlockable Achievements (aka kursmål/kunskapsmål)

Name	Short desc	Grade level	Assessment
A1	Konsekvent tillämpa procedurell abstraktion för att öka läsbarheten och undvika upprepningar	3	L
A2	Tillämpa objektorienterad abstraktion för att dölja implementationsdetaljer bakom väldefinierade gränssnitt	3	L
A3	Demonstrera förståelse för designprincipen informationsgömning i ett C-program med hjälp av .c och .h-filer	4	L, G
B4	Använda arv, metodspecialisering och superanrop i ett program som drar nytta av subtypspolymorfism	3	L

↑

ID

↑

Ingress

↑

Nivå

↑

Hur målet kan redovisas

3

L – Labb

4

W – Essä

5

T – Tobias

P – Presentation

R – Rapport



Mål

Unlockable Achievement (kursmål/kunskapsmål)

Name	Short desc
A1	Konsekvent tillämpa procedurell abstraktion för att öka läsbarheten och undvika upprepningar
A2	Tillämpa objektorienterad abstraktion för att dölja implementationsdetaljer bakom väldefinierade gränssnitt
A3	Demonstrera förståelse för designprincipen information hiding genom att implementera ett C-program med hjälp av .c och .h-filer
B4	Använda arv, metodspecialisering och superanrop för att lösa problem som drar nytta av subtypspolymorfism

(A. Abstraktion)

A1

Konsekvent tillämpa procedurell abstraktion för att öka läsbarheten och undvika upprepningar

Level Assessment

3 L

Abstraktion är en av de viktigaste programmeringsprinciperna. Vi vet att djupt, djupt nere under huven är allt bara ettor och nollor (redan detta är en abstraktion!), men ovanpå dessa har vi byggt lager på lager av abstraktioner som låter oss tala om program t.ex. i termer av strukturer och procedurer.

Proceduren `ritaEnCirkel(int radie, koordinat center)` utför beräkningar och tänder individuella pixlar på en skärm, men i och med att dessa rutiner kapslats in i en procedur med ett vettigt namn, där indata är uttryckt i termer av koordinater och radie har vi abstraherat bort dessa detaljer, och det blir möjligt att rita cirklar tills korna kommer hem utan att förstå hur själva implementationen ser ut.

Väl utförd abstraktion döljer detaljer och låter oss fokusera på färre koncept i taget.

Du bör ha en klar uppfattning om bland annat:

- Varför det är vettigt att identifiera liknande mönster i koden och extrahera dem och kapsla in dem i en enda procedur som kan anropas istället för upprepningarna?
- Abstraktioner kan "läcka". Vad betyder det och vad får det för konsekvenser?
- Vad är skillnaderna mellan "control abstraction" och "data abstraction"?
(Du kan läsa om dessa koncept på t.ex. [Wikipedia](#)).



Notera: målens namn/nummer kan ha ändrats

Redovisning

- **Din uppgift:** att övertyga examinatorn om att du uppfyller målet
 - Ge **exempel** från den kod (etc.) du har skrivit
 - Inga **core dumps**
 - Ha en **story** för hur allting hänger ihop
- Försök att alltid redovisa **> 1 mål åt gången**
 - Mindre arbete, mindre väntetid
 - Synergi!
- Demo av redovisningsystemet sker på föreläsning innan första redovisningstillfället



Var finns information om kursen?

The screenshot shows the IOOPM 2017 homepage with a yellow header and a city skyline background. It features a navigation bar with links like HOMEPAGE, DEADLINES, OM KURSEN, PROCESS, MÅL, UPPGIFTER, LÄNKAR, KONTAKT, and HJÄLP!. Below the header, there's a section titled 'KURSDELAR' with a box containing 'IMPERATIV OCH' and 'wrigstad.com/ioopm'. A yellow box highlights the URL 'wrigstad.com/ioopm'.

Hur interagerar jag med er?

The screenshot shows the Piazza platform interface. At the top, it says 'Class at a Glance' with statistics: 1 unread posts, 1 unanswered questions, and 8 unresolved followups. Below this is a 'Student Enrollment' bar. The main area lists various posts from students, such as 'Hjälp! Jag behöver någon att hjälpa', 'Hjälp! Hur kan man...?', and 'Hjälp! Kompletterat äpplet...'. On the right, there's a message from a professor: 'Last Fall, we launched a new service called Piazza Careers. And through this service, we have connected students with other students studying same subjects, with recent alumnae in industry, and with potential employers. Many of them have successfully secured internships and jobs through our service! Just as Piazza was born out of my desire to help my Computer Science class, Piazza Careers was born out of my desire to help my Computer Science students find internships and challenges faced greatest with what to do with why we live, helping enhance their experience in their chosen area of study. I am currently working on Piazza Careers to make it easier for students to connect with companies and with each other. We have a vision for Piazza Careers, and like with Piazza Q&A you can't get there without your feedback. We are eager to make a real impact on students' lives, helping enhance their experience in their chosen area of study. It will require a lot of iteration and hard work. We'll inevitably make a few mistakes along the way and we ask for your support and forgiveness in those moments. The Piazza Careers service is essentially a 'third rail' that students can choose to access when desired. Students can remove this additional tab at any time. Our foremost priority is to protect student privacy and keep their identities distinct from you and us.'

Hur lämnar jag in en uppgift?

The screenshot shows a GitHub repository page for 'IOOPM-UU / ioopm15'. It displays a list of commits, branches, and pull requests. A yellow box highlights the URL 'GitHub' at the bottom of the page.

Hur redovisar jag?

The screenshot shows the AU Portal interface with a section titled 'Unlockable Achievements (aka kursmål/kunskapsmål)'. It lists various achievements with descriptions, grades, and edit/delete buttons. A yellow box highlights the URL 'AU Portal' at the bottom of the page.

Name	Description	Grade level	Assessment	Show	Edit	Delete
A1	Konsekvent tillämpa procedurell abstraktion för att lösa läroböckens och undervisningsuppgifter	3	L	Show	Edit	Delete
A2	Tillämpa objektorienterad abstraktion för att dölja implementationsdetaljer bakom väldefinierade gränssnitt	3	L	Show	Edit	Delete
A3	Demonstrera förmåga för designprincipens informationsglänsning i ett C-program med hjälp av z och h-filer	4	L, G	Show	Edit	Delete
A4	Använda arb. metodspecialisering och superspecialisering i ett program som drar nytta av nättypolyforsamling	3	L	Show	Edit	Delete
A5	Använda både framförgrader och specialiserade konstruktorer på lämpligt sätt i programmering	3	L	Show	Edit	Delete
A6	Förklara hur användargruppen har använts i ett program till allt separat från kodkoden	4	G, T	Show	Edit	Delete
A7	Förklara designprinciper som visar hur man tillstyrkt designprinciper har använts i ett icke-trivialt program	4	G	Show	Edit	Delete
A8	Visa hur man kan separera gränssnitt från implementation med hjälp av Java-interfaces	4	L, G	Show	Edit	Delete
A9	Dokumentera lika tre olika metoder/generatorer så att någon utomstående kan prata med dem	3	L	Show	Edit	Delete
A10	Använda viss pekare relevant sätt, t.ex. av	3		Show	Edit	Delete
A11	Använda parametriskt utvecklade funktioner vid interaktion med Java	3		Show	Edit	Delete
A12	Designa med parametrar av ett program men så att det kan användas i flera olika situationer	3		Show	Edit	Delete
A13	Översätta mellan rekursiva och iterativa lösningar av samma problem samt diskutera till- och nackdelar med dessa tekniker	3	L	Show	Edit	Delete



Var finns information om kursen?

The screenshot shows the IOOPM 2017 homepage with a yellow header. Below it, there's a section titled "IMPERATIV OCH" with a link to "wrigstad.com/ioopm".

Hur interagerar jag med er?

The screenshot shows a Piazza class page with a "general" discussion thread. A separate window for "Rocket.Chat" is also visible.

Hur lämnar jag in en uppgift?

The screenshot shows a GitHub repository named "ioopm15" for the course IOOPM, 2015. It displays a list of commits and pull requests.

GitHub

Hur redovisar jag?

The screenshot shows the "AU Portal" with a section titled "Unlockable Achievements (aka kursmål/kunskapsmål)". It lists various achievements with descriptions and grades.

Name	Description	Grade level	Assessment
A1	Konsekvent tillämpa procedurell abstraktion för att lösa läsövningar och matematiska uppgifter	3	L
A2	Tillämpa objektorienterad abstraktion för att dölja implementationsdetaljer bakom väldefinierade gränssnitt	3	L
A3	Demonstrera förmåga för designprincipens informationsglänsning i ett C-program, med hjälp av c och h-filer	4	L, G
A4	Använda arb., metodspecialisering och supersköring i ett program som drar nytta av nättypolyforsim	3	L
A5	Använda både överbryggar och specialiserade konstruktörer på lämpligt sätt i programmering	3	L
A6	Förklara hur användningsgrupp har använts i ett program för att separera grönmarkering från koden	4	G, T
A7	Förklara hur designmedel som visar hur man kan designa och implementera en klass har använts i ett icke-trivialt program	4	G
A8	Visa hur man kan separera gränssnitt från implementation med hjälp av Java-interfaces	4	L, G
A9	Dokumentera lika tre olika metoder för att göra ett utomstående konsistenter med dess källkod	3	L
A10	Använda viss pekare relevant sätt, t.ex. av en struktur vid iteration medan man inte är i den strukturen	3	L
A11	Använda parametrisk kod för att generera kod vid interaktion med en lista	3	L
A12	Designa med parametrar av ett program mer än en gång	3	L
A13	Översätta mellan rekursiva och iterativa lösningar av samma problem samt diskutera till- och nackdelar med olika tekniker	3	L

AU Portal



Du börjar här:

The screenshot shows a web browser window with the URL `wrigstad.com` in the address bar. The main title is **IOOPM 2017** with the subtitle **IMPERATIV ... OCH SÅ VIDARE**. Below the title is a navigation menu with links: SCHEMA (highlighted in dark grey), DEADLINES, HUR FUNKAR IOOPM?, PROCESS, PROGRAMMERA, MÅL, KODPROV, LÄNKAR, KONTAKT, and HJÄLP!. The background features a blurred image of a city skyline at sunset. On the left side of the page, there is a sidebar with the heading **LÄNKAR** and links to **Inlämningsuppgifter** and **AU Portal**. At the bottom of the page is a call-to-action button with the text **wrigstad.com/ioopm**.

SCHEMA DEADLINES HUR FUNKAR IOOPM? PROCESS PROGRAMMERA MÅL
KODPROV LÄNKAR KONTAKT HJÄLP!

LÄNKAR

Inlämningsuppgifter

AU Portal

Visa en meny

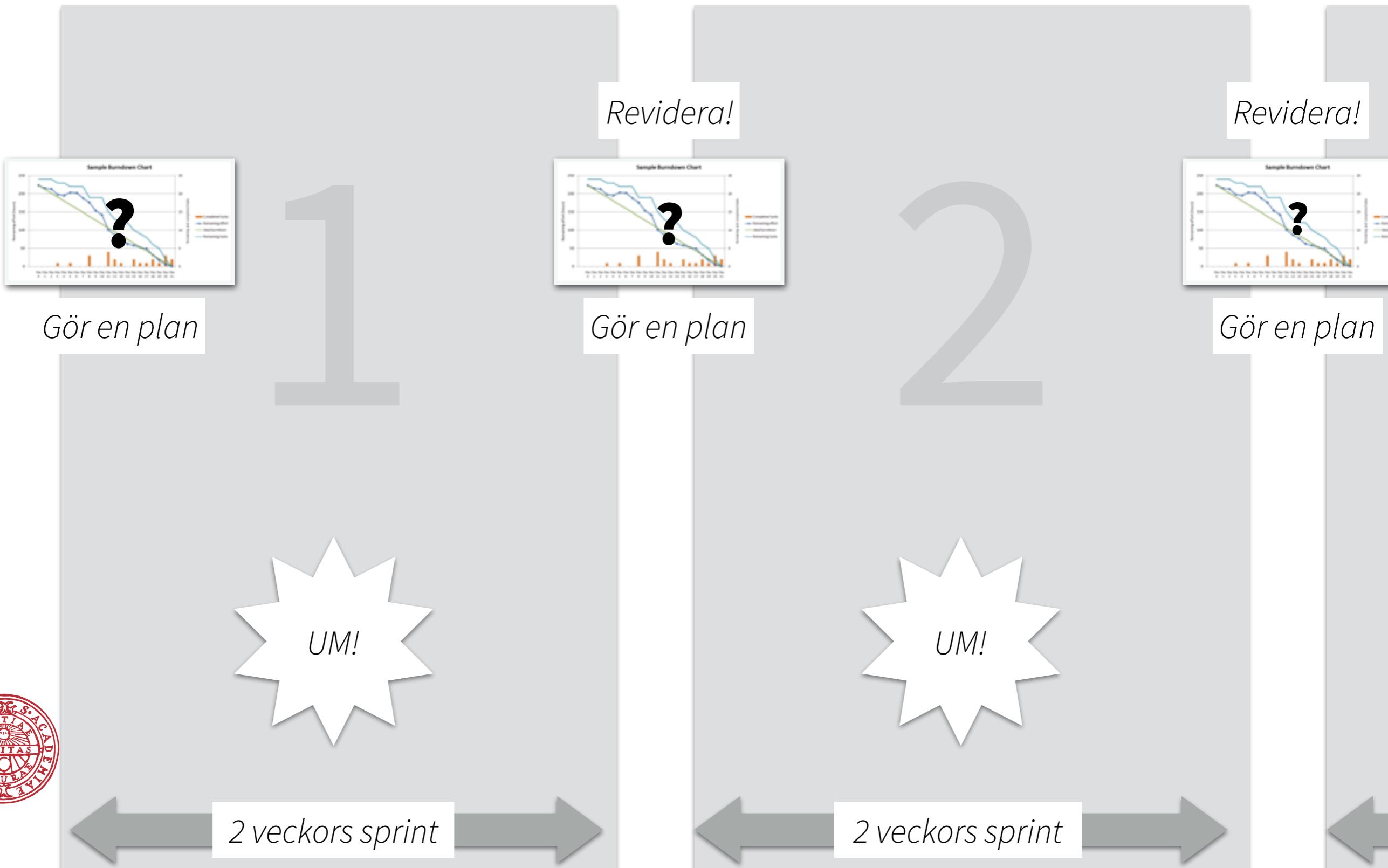
wrigstad.com/ioopm



Översikt

Fas 1 (Sep–Okt)					Fas 2 (Okt–Nov)				Fas 3 (Dec–Jan)		
Imperativ programmering i C					OO med Java				Projektet		
<i>Intro</i>	<i>Intro</i>	<i>Sprint 1</i>	<i>Sprint 2</i>	<i>Kodprov</i>	<i>Intro</i>	<i>Sprint 1</i>	<i>Sprint 2</i>	<i>Kodprov</i>	<i>Fria sprintar</i>		
		UM 1	UM 2			UM 4	UM 5				
		Lager	Lager Resepl.			Kassakö Twitterish	MUD Kalkylator		<i>UM = uppföljningsmöte</i>		
14 föreläsningar 15 labbar			10 föreläsningar 10 labbar				1 föreläsning 4 labbar (initialt)				
<i>Deadlines</i>											

Fas [kursen har 3]



Sprint [varje fas har 2]

Uppgift

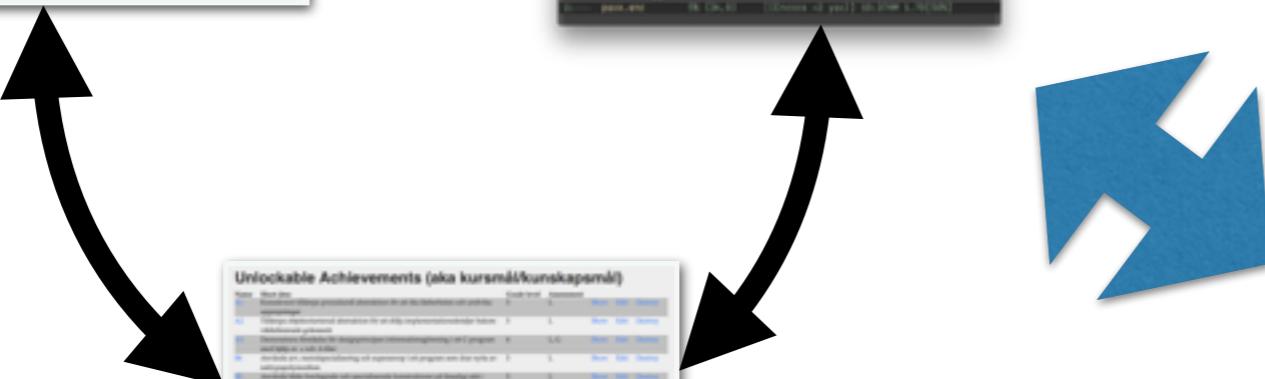
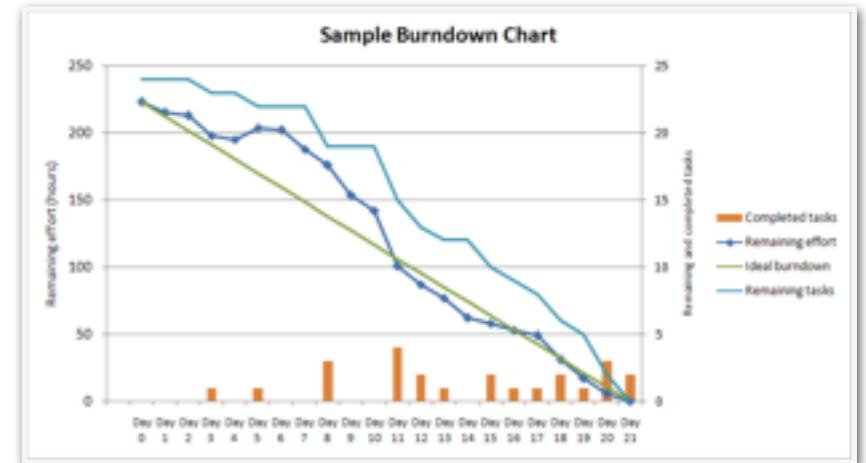


Implementera

```
embed void
    this->nodes = callc(k, sizeof(node_t));
end

def insert(vint) : bool {
    embed void
    node_t **n = (node_t**) &this->root;
    while (*n)
    {
        node_t *c = *n;
        if (c->value == v) { return false; }
        n = (c->value < v) ? &(c->left) : &(c->right);
    }
    ((node_t*)this->nodes)[this->size++] = (node_t) {
        .value = v;
        *n = &((node_t*)this->nodes)[this->size-1];
    };
    end;
    else;
}
def reset() : void
{
    pass;
}
```

För bok över dina framsteg



Välj mål



Redovisa

Uppföljningsmöte

- I slutet av varje sprint

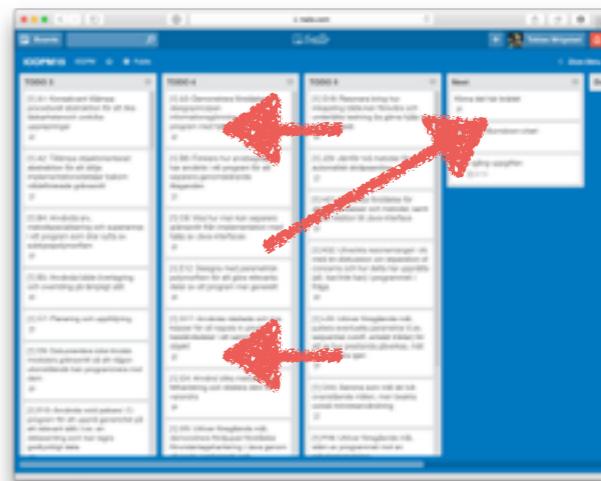
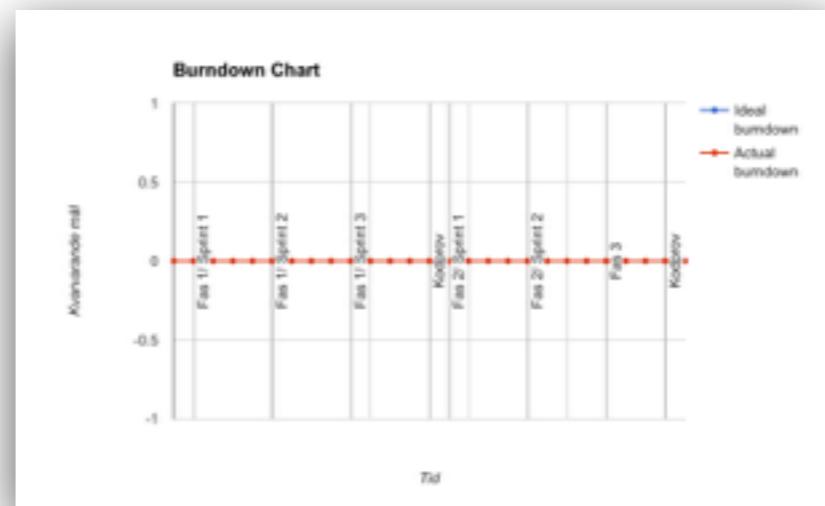
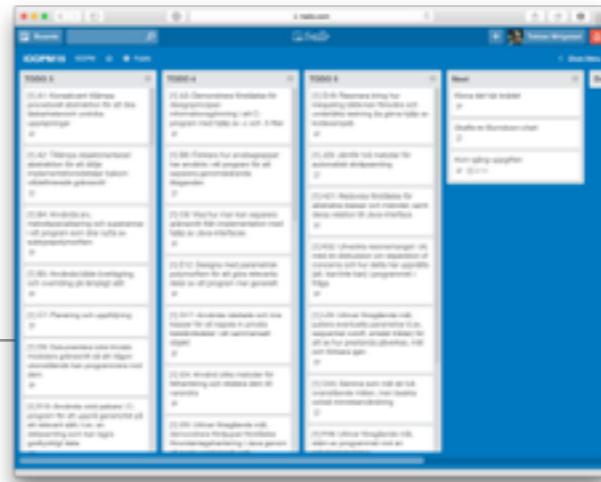
Diskussion med assistent
utifrån burndown chart

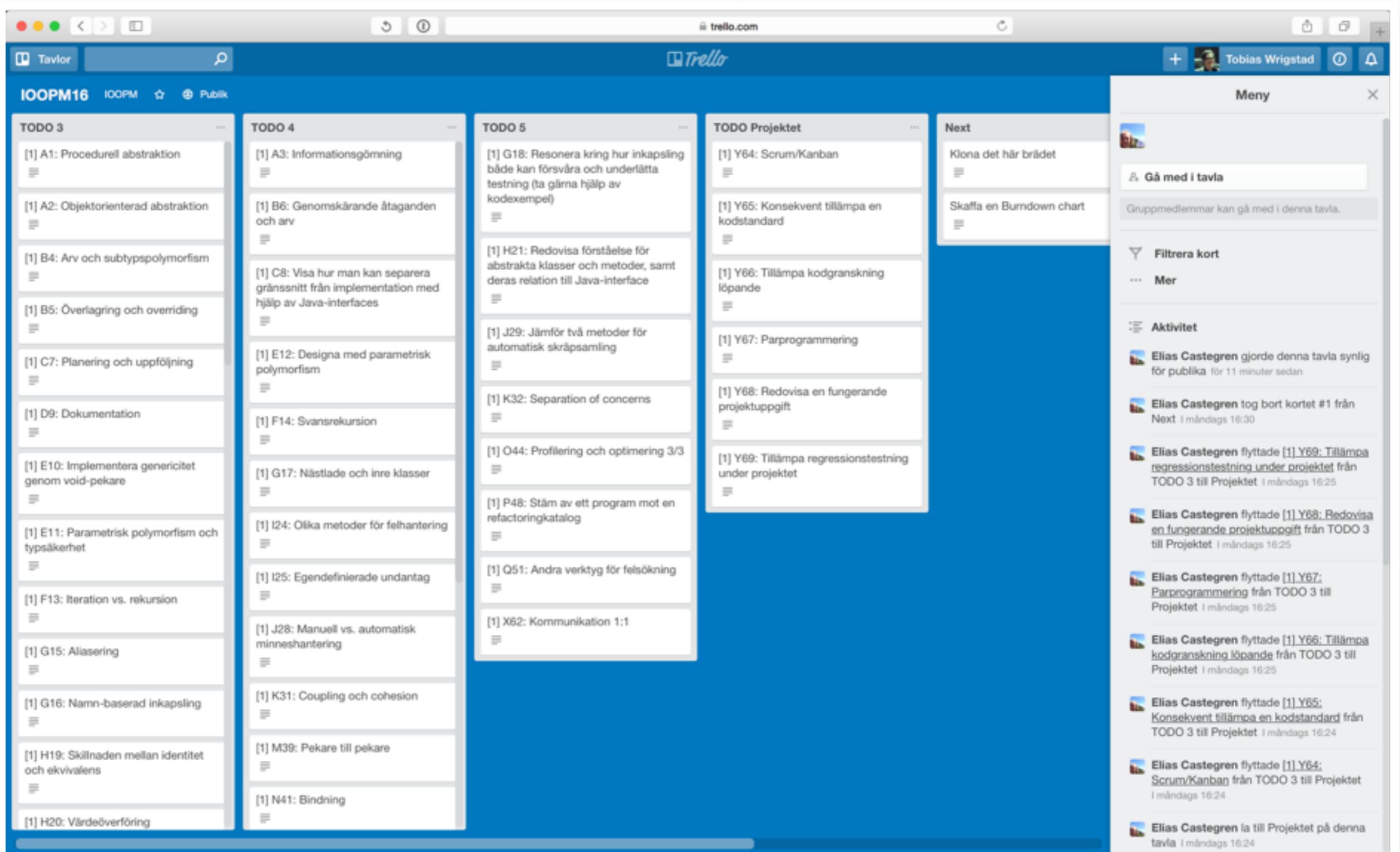
Tillsammans med tre
andra studenter

I början av kursen är det
fokus på planering

Vi går mot fokus på
uppföljning

- Nästa möte nu på fredag




 A screenshot of a Trello board titled "IOOPM16". The board has five columns: "TODO 3", "TODO 4", "TODO 5", "TODO Projektet", and "Next". Each column contains a list of tasks, many of which are preceded by a small icon like a file or a person. The "TODO Projektet" column contains tasks related to project management like "Scrum/Kanban" and "Parprogramming". The "Next" column contains tasks for cloning the board and creating a Burndown chart. The right side of the screen shows a sidebar with a "Meny" button, a user profile for Tobias Wrigstad, and a list of recent activities.

Column	Tasks
TODO 3	[1] A1: Procedurell abstraktion [1] A2: Objektorienterad abstraktion [1] B4: Arv och subtypspolymorfism [1] B5: Överlagring och overriding [1] C7: Planering och uppföljning [1] D9: Dokumentation [1] E10: Implementera genericitet genom void-pekarer [1] E11: Parametrisk polymorfism och typesäkerhet [1] F13: Iteration vs. rekursion [1] G15: Aliasering [1] G16: Namn-baserad inkapsling [1] H19: Skillnaden mellan identitet och ekvivalens [1] H20: Värdeöverföring
TODO 4	[1] A3: Informationsgömning [1] B6: Genomskärande åtaganden och arv [1] C8: Visa hur man kan separera gränssnitt från implementation med hjälp av Java-interfaces [1] E12: Designa med parametrisk polymorfism [1] F14: Svansrekursion [1] G17: Nästlade och inre klasser [1] I24: Olika metoder för felhantering [1] I25: Egendefinierade undantag [1] J28: Manuell vs. automatisk minneshantering [1] K31: Coupling och cohesion [1] M39: Pekare till pekarer [1] N41: Bindning
TODO 5	[1] G18: Resonera kring hur inkapsling både kan försvåra och underlättा testning (ta gärna hjälp av kodexempel) [1] H21: Redovisa förståelse för abstrakta klasser och metoder, samt deras relation till Java-interface [1] J29: Jämför två metoder för automatisk skräpsamling [1] K32: Separation of concerns [1] O44: Profilering och optimering 3/3 [1] P48: Stäm av ett program mot en refactoringkatalog [1] Q51: Andra verktyg för felsökning [1] X62: Kommunikation 1:1
TODO Projektet	[1] Y64: Scrum/Kanban [1] Y65: Konsekvent tillämpa en kodstandard [1] Y66: Tillämpa kodgranskning löpande [1] Y67: Parprogrammering [1] Y68: Redovisa en fungerande projektuppgift [1] Y69: Tillämpa regressionstestning under projektet
Next	Klona det här brädet Skaffa en Burndown chart



Uppgifter | Imperativ och objektorienterad... Faser | Imperativ och objektorienterad... Futures: improvements on UpScale... Creating cards by email - Trello Help +

Boards UpScale Doing Implement dependencies Basic data structures Module system Add a card...

Futures: improvements in list [Next](#) [Edit](#)

Members: AN +

Description [Edit](#)
Subsumes the following (archived) cards:

- [Await & Suspend](#)
- [Future chaining](#)
- [Futures](#)
- [Coroutines](#)
- [Suspendable/blocking actors \(was Futures etc.\)](#)

Checklist [Delete...](#)

0%

- Merge "children" and "responsibility" in the future struct
- Trace functions for futures and future type struct
- Remove stupid limitations (like 16 responsibilities max) on futures
- Add comprehensive testing for non-deterministic behaviour on futures, chaining, await and suspend

Add an item...

Activity

 Write a comment...

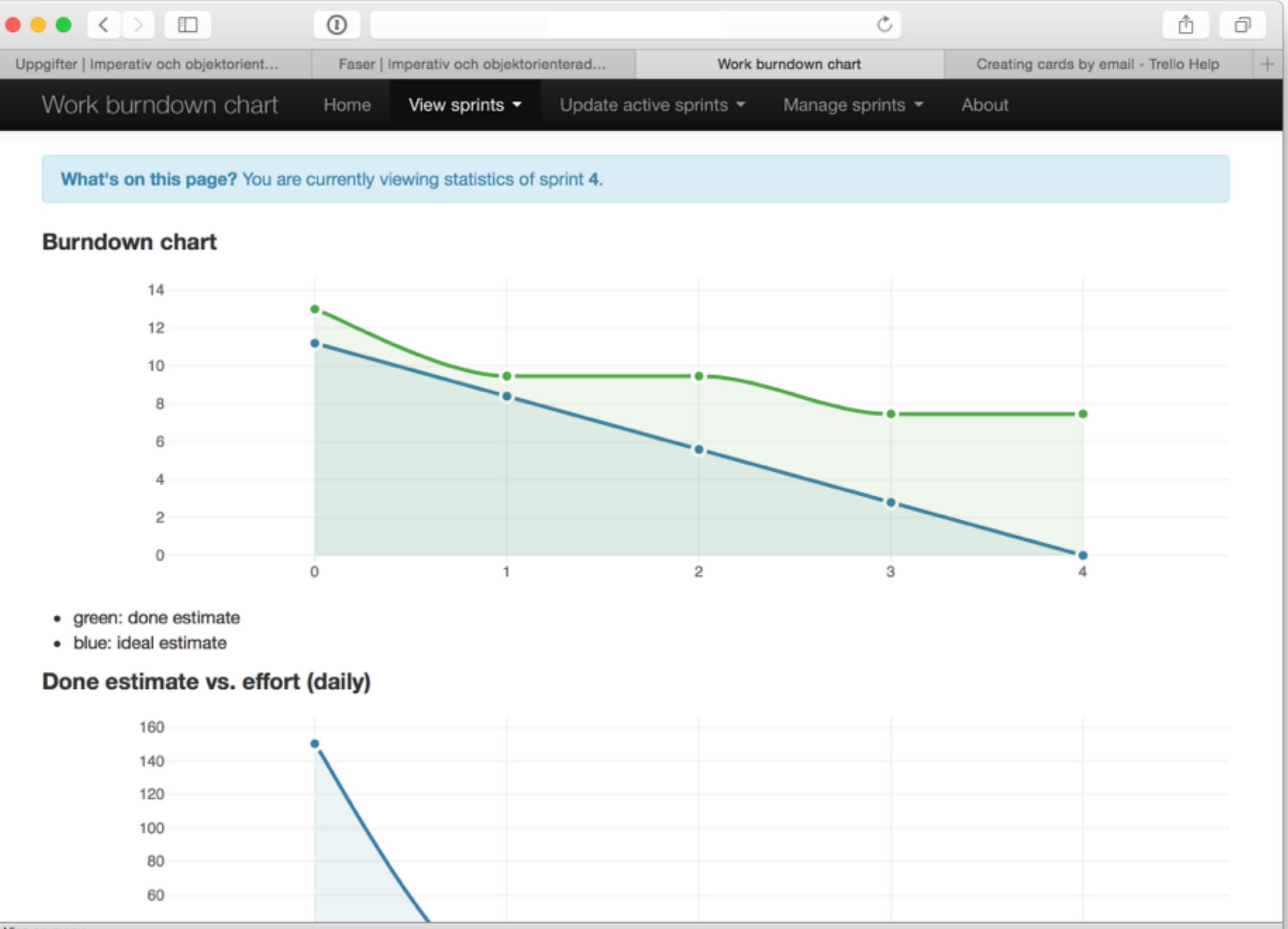
Add

- Members
- Labels
- Checklist
- Due Date
- Attachment

Actions

- Move
- Copy
- Subscribe
- Archive

[Share and more...](#)



Visa en meny

<https://github.com/devtyr/trello-burndown>

docs.google.com

tobias.wrigstad@gmail.com

Comments Share

Burndown chart

File Edit View Insert Format Data Tools Add-ons Help All changes saved in Drive

B C D E F G H I J K

1

2 Hur många mål siktar du på att ta denna sprint? 7

3

4

5 Hur många mål har du tagit?

	Lab 2	1
	Lab 3	0
	Lab 4	3
	Lab 5	3

6

7

8

9

10

11

12

13

14

15

16

17

18

19

20

21

22

23

24

25

26

27

28

29

30

Burndown Chart (Fas 1 / Sprint 1)

Ideal velocit

Actual velocit

Kvarvarande mål

Tid

The chart displays two lines: a blue line for 'Ideal velocit' and a red line for 'Actual velocit'. The y-axis represents 'Kvarvarande mål' (Remaining tasks) from 0 to 8. The x-axis represents 'Tid' (Time). The ideal velocity is a straight line starting at approximately (0, 7.2) and ending at (5, 0). The actual velocity starts at (0, 7.2), dips slightly, then fluctuates between 6 and 6.5 tasks remaining until about time 4, after which it drops sharply to 0 by time 5.

Burndown totalt Fas1/Sprint1 Fas1/Sprint2 Fas1/Sprint3 Fas2/Sprint1 Fas2/Sprint2 Fas3 Burndown

Visa en meny

Se "Länkar" på kurswebben

Högskolepoäng

	HP	Deadline
Fas 1	5	V44*
Fas 2	5	V49
Fas 3	5	V2
Kodprov	2,5	17/10
	2,5	21/12

*It's complicated (se kurswebben för detaljer)



Övning i skriftlig färdighet

- På nivå 4 och 5 måste du redovisa ett mål som en essä
(För att nå nivå 3 räcker det med projektrapporten)
- Instruktioner finns på kurswebben

Omfattning: 7500 tecken

Deadline: 5/12

- Lämnas in via GitHub

ng till handledning online (t.ex. via epost)	11	13.4%
Återkoppling/feedback på inlämningar	24	29.3%
(utveckla gärna i kommentarerna nedan)	2	2.4%

Jämförelse mellan två
skräpsamlingsalgoritmer
Mark and Sweep mot Reference counting

Uppsala universitet
January 16, 2015

Två skräpsamlingsmetoder

Något som blir vanligare och vanligare är användningen av skräpsamlare, även känd som garbage collectors. En skräpsamlares främsta uppgift är att lämna tillbaka minne du inte använder. Så den frigör minne som du använt och inte använder längre, på ett automatiskt sätt. Denna process brukar kallas för skräpsamling. Anledningen till användningen av skräpsamlare har blivit stort skulle kunna bero på språk såsom Java, JavaScript, Python som alla använder sig av någon typ av skräpsamling. Varför beslutade sig folk för att använda skräpsamlare? Förmodligen för att det är svårt att hantera minnet manuellt. Om inte programmeraren har skickligheten som krävs och är på sin väkt hela tiden kan det uppstå minnesstöcker eller andra problem vid manuell hantering av minne. Med en skräpsamlan behöver inte programmeraren orska sig över sådana saker och kan ägna mer tid till andra saker.

Jag är ganska övertygad att du någon gång kommer använda dig av någon skräpsamlare om du ångar dig åt programering. Hur mycket du tänker på det eller inte är en annan fråga. Jag kommer beskriva hur två metoder för skräpsamling går till och göra en jämförelse mellan dem.

I detta dokument ska jag beskriva två vändiga algoritmer för skräpsamling och deras skillnader. Metoderna jag kommer fokussera på är Mark and Sweep och Reference counting.

Om ett objekts referensräknare sätter till si finns det inte längre några referenser till det objekten, Objekten är därmed skräp och kommer att frigöra direkt sär referensräknande skräpsamlnare är därmed deterministisk², vilket innebär att vi vet exakt när ett objekt tas bort. Detta är en av de stora fördelarna med referensräkning. Detta innebär att referensräkning är kompatibel med MUTEX och kan därmed användas tillsammans med destruktören, kod som körs automatiskt när objekten dörs.

En annan fördel med referensräkning är att arbetet för att ta reda på vad som är skräp är fördelat över hela programmet istället än föret med spårande skräpsamlnare. Detta är särskilt bra av utgående med referensräkning eftersom det kräver en liten del extra arbete borta för att uppdatera objekts referensräknare.³ Men ju före en referens räknas eller ändras en objekts referens lättas upp i minnet och dess referensräknare uppdateras. Det kan medföra stora mängder missar när objekten lättas upp i cache/minnet, vilket påverkar prestandan negativt.

Det kanske största problemet med referensräknande skräpsamlnare är att några implementeringar inte kan hantera cirkulära strukturer. Cirkulära strukturer innehåller objekt som direkt eller indirekt refererar till sig självt. Några implementeringar klarar detta genom att använda svaga referenser som inte enbart inte klar objekts referensräkningen. Det finns också mer komplexa algoritmer som kan hantera detta, men dessa kräver ofta stora mängder extra arbete.

2.2 Mark and sweep

Mark and sweep tillhör instugrön av tracing eller spårande skräpsamlnare. Denna typ av skräpsamling angriper problemet från Mark-and-Sweep. Istället för att hålla reda på och frigöra skräp direkt när det uppstår så kommer skräpsamlingens sätt vid olika trappsteg. Ofta startar detta när mängden ledigt minne tar slut eller faller under en viss nivå.

Mark and sweep har två steg som här kallas för "mark" och "sweep". Första stegen mark, gör ut på att hitta och markera alla levande objekt. Själva markeringen sätter genomsatt en flagga som sparar tillsammans med varje objekts sätta. Ista "mark" steget påbörjas nödsett samtliga flaggar.

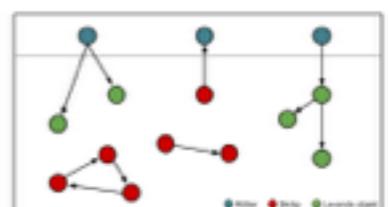


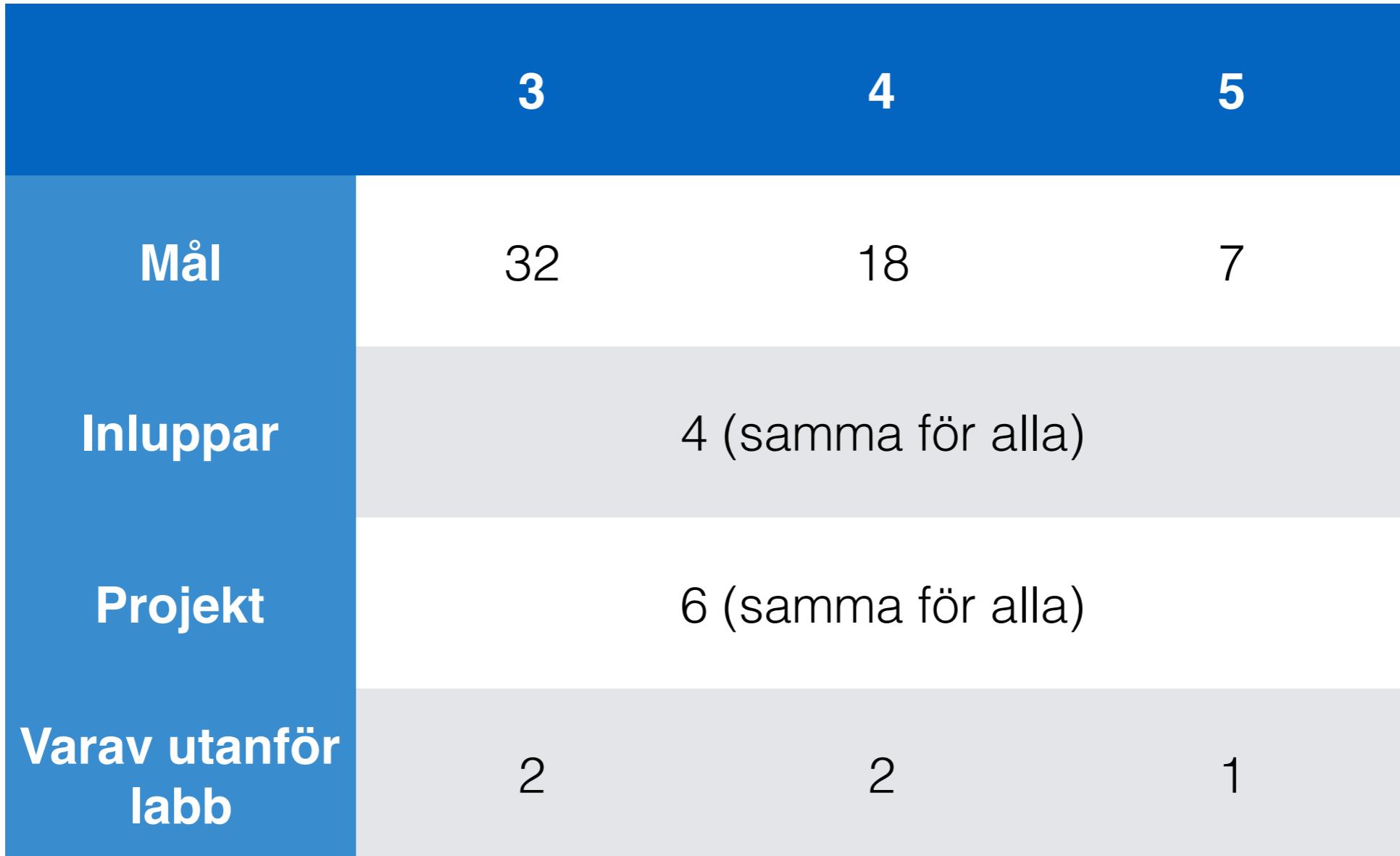
Figure 1: Objekter representerar objekt och pilarna referenser.

För att hitta alla levande objekt är unga skräpsamlare från ett antal rötter. Rötterna är alda referenser eller pelare som vi vet inte vi har tillträffat till. I programmet är just dessa varia alda pelare som tappar på staden. Skräpsamlaren går över rötterna och tifjer dessa referenser. Varje objekt som hittas markeras och för objekts senliga referenser är upprepat numera process som för släckning. Alltså vi tifjar referenser och alla objekt som hittas markeras och dessa referenser färs. Denna visualiseras i figuren.

²Detta är inte helt sant, då i faktiskt program kan uppdateringen av referensräkningen kan hända till och med.



Betyg



Fas 3: Projektarbete

- Arbeta 4–6 personer (vi tar fram grupperna under november)
- Uppgiften TBA (blir variant av tidigare år)

Rekommenderad start: 1/12

Klart: 9/1

- Lämnas in via GitHub
- Presentation och verkstad med annan grupp
- Grupperna lägger själva upp sprintar
- 1–2 KLOC, plus tester

3 *Uppgiften*

Uppgiften går ut på att utveckla ett bibliotek, för enkeltets skull kallat "gc", för minneshantering i form av en kompakterande skräpsamlares. Med funktionen `b_init` kan en användare reservera en egen "heap" – ett konsekutivt minnesblock¹ i vilket man sedan kan allokerar minne med hjälp av biblioteksfunktioner. Detta minne ska sedan hanteras automatiskt – när minnet tas slut ska skräpsamlingen automatiskt triggas, och alla objekt i detta minne som inte är nödvändigtvis rot i systemet tas bort².

Av pedagogiska skull beskrivs vi först skräpsamling med hjälp av mark-sweep, som vi öfver skulle använda innan vi går in på den kompakterande skräpsamlaren som använder en liknande algoritm.

3.1 Skräpsamling med mark-sweep

Skräpsamling med mark-sweep vandrar genom (traverserar) den graf som heapen utgör för att identifiera objekt som säkert kan destrueras utan att programmet kraschs. Vi går igenom algoritmen steg-för-steg nedan.

Vi kan tänka om att varje objekt innehåller en extra bit³, den s.k. mark-biten. När denna bit är satt (1) anses objektet vara "vid liv". Ansas är objektet skräp som kan tas bort.

Vid skräpsamling sker följande (logiskt sett):

Steg 1 Iterera över samtliga objekt på heapen och sätt mark-biten till 0. Detta innebär att alla objekt anses vara skräp initialt.

Steg 2 Sök igenom stacken eller pekare till objekt på heapen⁴, och med utgångspunkt från dessa objekt, traversera heapen och markera alla objekt som påträffats genom att mark-biten sätts till 1.

Steg 3 Iterera över lista över samtliga objekt på heapen och frigör alla objekt vars mark-bit fortfarande är 0.

Steg 2 kallas för "mark-fasen" och steg 3 för "sweep-fasen", härav algoritmens namn, mark-sweep.

OBSERVERA
Denna del av specificeringen är ett
livsmedel dokument som kan kom-
ma att uppdateras och förändras
under projektets gång.

¹T.ex. med hjälp av `malloc` i
`stdlib.h`, eller `new` i `new.h`.

²Vi gör en förenklning och utgår från
att programmet är enkelträddade
och att endast en heap skapas per
program.

³Tekniskt kan det också vara en bit
som man har en över. Dådand kan man
packa in bitar i annat data – vi skall
se exempel på det senare i denna text!

⁴Dessa pekare kallas vi också för
"vötter".



Kodprovet (2x2,5 HP)

- Två frågor – en C, en Java

Individuellt prov i datorsal, 3 timmar – **ingen tillgång till Internet**

- Syftet: att tvinga alla att sitta i framsätet vid parprogrammering

Examinerar inte kursmål!

- Går att ta i steg (klara en fråga på varje prov)

- Tre provtillfällen under kursen

17/10 och 21/12 och 5/1

- Anmälan annonseras i Piazza



Simple [minimal sammanfattning]

1. Läs specifikationen och leta specifikt efter **verb** (funktioner/beteende), eller **substantiv** (data/objekt/strukturer) — gör en work breakdown structure
2. Skriv kod för att pröva om du tänkt rätt (vad är rätt – hur man kollar det?)
3. Ha alltid ett fungerande program
4. Kompilera efter varje förändring
5. Kör programmet hela tiden för att hitta fel (eller ännu bättre – kör testen!)
6. Dela upp alla problem i delproblem, gå till 7. först när något är enkelt
7. Dela upp alla delproblem i mindre steg – gör de enklaste först
8. **Fuska** (cheat) varje gång du riskerar att fastna
9. **Skarva** (dodge) för att förenkla specifikationer och skapa fler enklare delsteg
10. Växla mellan att: tänka, koda och ibland refaktorera (speciellt **fusk** och **skarvar**)



Simple

- Om du inte redan är en programmerare måste du använda SIMPLE under kursen

- Finns detaljerad beskrivning på kurswebben

Använder Lab 4, 5 och inlupp 1 som löpande exempel

- Det kan vara svårt att ta in allt direkt, så börja med det som verkar enkelt

Försök inte göra rätt, utan det som känns rätt — gå tillbaka till texten när det behövs



Att använda en texteditor

- Under kursen kommer vi att använda **Emacs**

Under fas 2 är de tillåtet att använda IDE:er, men inte rekommenderat

- Emacs kan också betyda vim men **inte**

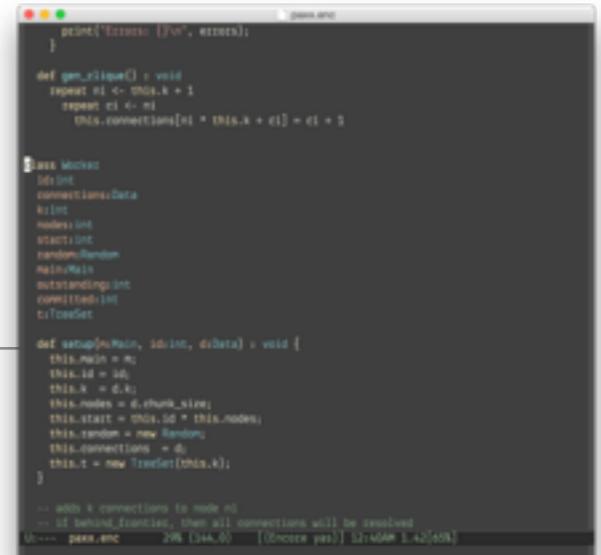
Gedit

Nano

Notepad++

Sublime

•



Sammanfattningsvis

- Från och med nästa vecka skall du jobba med inlämningsuppgifterna
 - Jobba i ett programmeringspar
- Labbarna är till för redovisning och hjälp och är **inte obligatoriska**
- Kursen kretsar till 75% kring redovisning av **mål** som finns beskrivna på kursens webbsida
- Alla mål redovisas i par men betygsätts individuellt
- Du förväntas själv göra kopplingen mellan mål och uppgifter
 - Viss handledning finns i form av tips i uppgiftstexterna
- **Implementera först, redovisa sedan**