

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 2016 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'. Another yellow box highlights the 'NYHETER' section.

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 shows a list of posts from students, with one post highlighted in yellow. A yellow box highlights the word 'Piazza'.

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 repository name 'ioopm15'.

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 title 'Unlockable Achievements'.



Var finns information om kursen?

The screenshot shows the IOOPM 2016 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 interface with a sidebar and a main feed. A yellow box highlights the "Rocket.Chat" window, which displays a list of messages.

Hur lämnar jag in en uppgift?

The screenshot shows a GitHub repository page for "ioopm15". It includes 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.

AU Portal



Du börjar här:

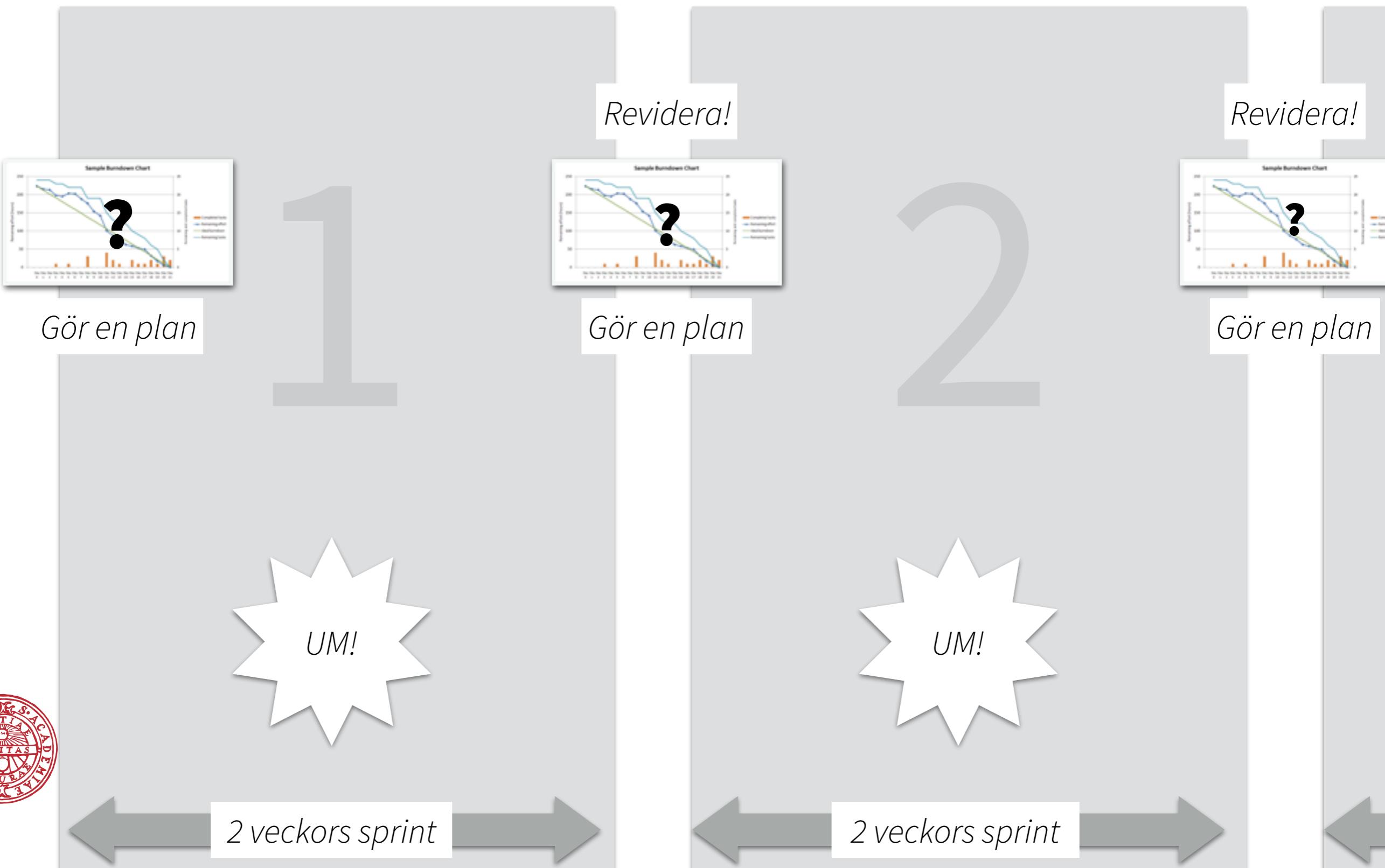
The screenshot shows a web browser window with the URL wrigstad.com in the address bar. The main title is **IOOPM 2016** with the subtitle **IMPERATIV ... OCH SÅ VIDARE**. Below the title is a navigation menu with links: SCHEMA (highlighted in dark grey), DEADLINES, HUR FUNKAR IOOPM?, PROCESS, PROGRAMMERA, 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 sidebar is a button labeled [Visa en meny](#). To the right of the sidebar is a large central text block with the title **IMPERATIV OCH
OBJEKTORIENTERAD
PROGRAMMERINGS-METODIK** and a call-to-action button with the text 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

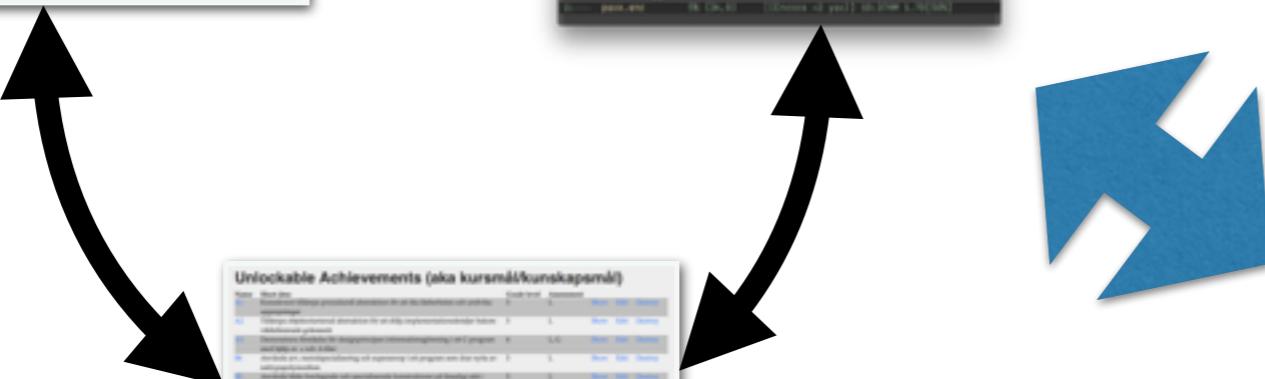
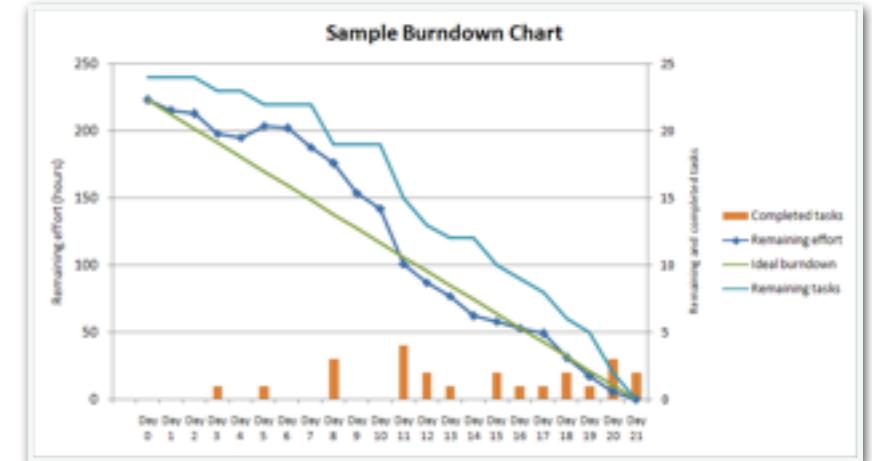


Implementera

```
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

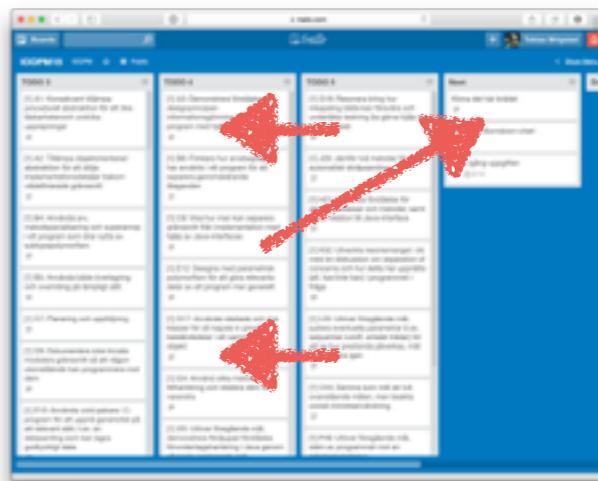
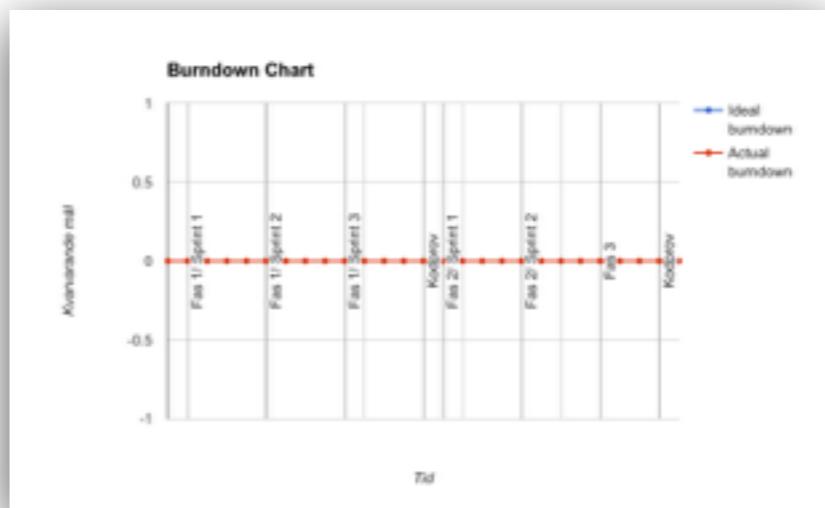
Diskussion med assistent
utifrån burndown chart

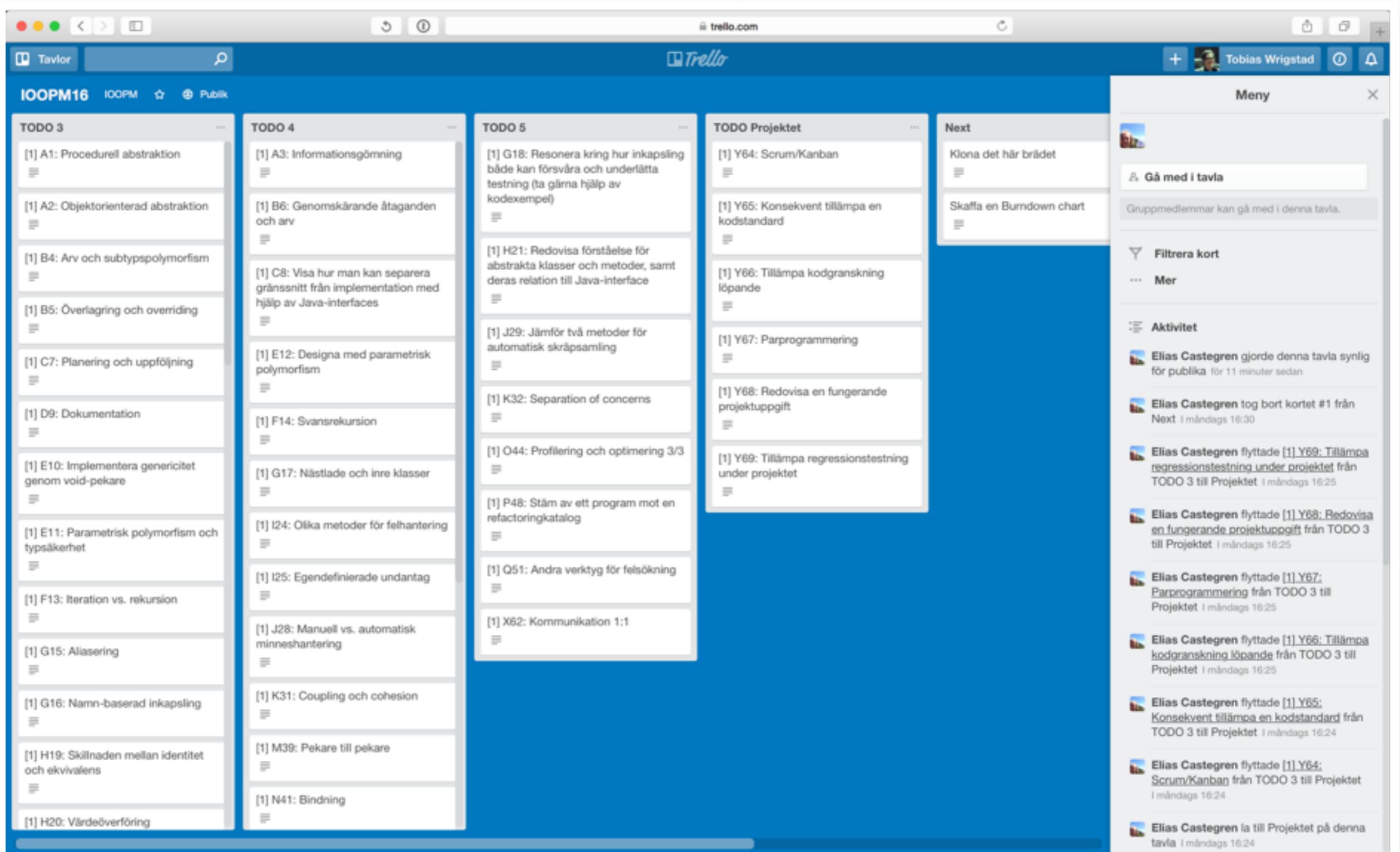
Tillsammans med tre
andra studenter

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 "[1]". The tasks are organized into rows, with some rows spanning multiple columns. The "Next" column contains two items: "Klona det här brädet" and "Skaffa en Burndown chart". On the right side of the board, there is a sidebar with a "Meny" button, a user profile for "Tobias Wrigstad", and a list of recent activities. The activities log shows various actions taken by "Elias Castegren" such as making the board public, removing cards, and moving them between columns.

Column	Task 1	Task 2	Task 3	Task 4	Task 5	Task 6	Task 7	Task 8	Task 9	Task 10	Task 11	Task 12	Task 13	Task 14	Task 15	Task 16	Task 17	Task 18	Task 19	Task 20	Task 21	Task 22	Task 23	Task 24	Task 25	Task 26	Task 27	Task 28	Task 29	Task 30	Task 31	Task 32	Task 33	Task 34	Task 35	Task 36	Task 37	Task 38	Task 39	Task 40	Task 41	Task 42	Task 43	Task 44	Task 45	Task 46	Task 47	Task 48	Task 49	Task 50	Task 51	Task 52	Task 53	Task 54	Task 55	Task 56	Task 57	Task 58	Task 59	Task 60	Task 61	Task 62	Task 63	Task 64	Task 65	Task 66	Task 67	Task 68	Task 69	Task 70	Task 71	Task 72	Task 73	Task 74	Task 75	Task 76	Task 77	Task 78	Task 79	Task 80	Task 81	Task 82	Task 83	Task 84	Task 85	Task 86	Task 87	Task 88	Task 89	Task 90	Task 91	Task 92	Task 93	Task 94	Task 95	Task 96	Task 97	Task 98	Task 99	Task 100	Task 101	Task 102	Task 103	Task 104	Task 105	Task 106	Task 107	Task 108	Task 109	Task 110	Task 111	Task 112	Task 113	Task 114	Task 115	Task 116	Task 117	Task 118	Task 119	Task 120	Task 121	Task 122	Task 123	Task 124	Task 125	Task 126	Task 127	Task 128	Task 129	Task 130	Task 131	Task 132	Task 133	Task 134	Task 135	Task 136	Task 137	Task 138	Task 139	Task 140	Task 141	Task 142	Task 143	Task 144	Task 145	Task 146	Task 147	Task 148	Task 149	Task 150	Task 151	Task 152	Task 153	Task 154	Task 155	Task 156	Task 157	Task 158	Task 159	Task 160	Task 161	Task 162	Task 163	Task 164	Task 165	Task 166	Task 167	Task 168	Task 169	Task 170	Task 171	Task 172	Task 173	Task 174	Task 175	Task 176	Task 177	Task 178	Task 179	Task 180	Task 181	Task 182	Task 183	Task 184	Task 185	Task 186	Task 187	Task 188	Task 189	Task 190	Task 191	Task 192	Task 193	Task 194	Task 195	Task 196	Task 197	Task 198	Task 199	Task 200	Task 201	Task 202	Task 203	Task 204	Task 205	Task 206	Task 207	Task 208	Task 209	Task 210	Task 211	Task 212	Task 213	Task 214	Task 215	Task 216	Task 217	Task 218	Task 219	Task 220	Task 221	Task 222	Task 223	Task 224	Task 225	Task 226	Task 227	Task 228	Task 229	Task 230	Task 231	Task 232	Task 233	Task 234	Task 235	Task 236	Task 237	Task 238	Task 239	Task 240	Task 241	Task 242	Task 243	Task 244	Task 245	Task 246	Task 247	Task 248	Task 249	Task 250	Task 251	Task 252	Task 253	Task 254	Task 255	Task 256	Task 257	Task 258	Task 259	Task 260	Task 261	Task 262	Task 263	Task 264	Task 265	Task 266	Task 267	Task 268	Task 269	Task 270	Task 271	Task 272	Task 273	Task 274	Task 275	Task 276	Task 277	Task 278	Task 279	Task 280	Task 281	Task 282	Task 283	Task 284	Task 285	Task 286	Task 287	Task 288	Task 289	Task 290	Task 291	Task 292	Task 293	Task 294	Task 295	Task 296	Task 297	Task 298	Task 299	Task 300	Task 301	Task 302	Task 303	Task 304	Task 305	Task 306	Task 307	Task 308	Task 309	Task 310	Task 311	Task 312	Task 313	Task 314	Task 315	Task 316	Task 317	Task 318	Task 319	Task 320	Task 321	Task 322	Task 323	Task 324	Task 325	Task 326	Task 327	Task 328	Task 329	Task 330	Task 331	Task 332	Task 333	Task 334	Task 335	Task 336	Task 337	Task 338	Task 339	Task 340	Task 341	Task 342	Task 343	Task 344	Task 345	Task 346	Task 347	Task 348	Task 349	Task 350	Task 351	Task 352	Task 353	Task 354	Task 355	Task 356	Task 357	Task 358	Task 359	Task 360	Task 361	Task 362	Task 363	Task 364	Task 365	Task 366	Task 367	Task 368	Task 369	Task 370	Task 371	Task 372	Task 373	Task 374	Task 375	Task 376	Task 377	Task 378	Task 379	Task 380	Task 381	Task 382	Task 383	Task 384	Task 385	Task 386	Task 387	Task 388	Task 389	Task 390	Task 391	Task 392	Task 393	Task 394	Task 395	Task 396	Task 397	Task 398	Task 399	Task 400	Task 401	Task 402	Task 403	Task 404	Task 405	Task 406	Task 407	Task 408	Task 409	Task 410	Task 411	Task 412	Task 413	Task 414	Task 415	Task 416	Task 417	Task 418	Task 419	Task 420	Task 421	Task 422	Task 423	Task 424	Task 425	Task 426	Task 427	Task 428	Task 429	Task 430	Task 431	Task 432	Task 433	Task 434	Task 435	Task 436	Task 437	Task 438	Task 439	Task 440	Task 441	Task 442	Task 443	Task 444	Task 445	Task 446	Task 447	Task 448	Task 449	Task 450	Task 451	Task 452	Task 453	Task 454	Task 455	Task 456	Task 457	Task 458	Task 459	Task 460	Task 461	Task 462	Task 463	Task 464	Task 465	Task 466	Task 467	Task 468	Task 469	Task 470	Task 471	Task 472	Task 473	Task 474	Task 475	Task 476	Task 477	Task 478	Task 479	Task 480	Task 481	Task 482	Task 483	Task 484	Task 485	Task 486	Task 487	Task 488	Task 489	Task 490	Task 491	Task 492	Task 493	Task 494	Task 495	Task 496	Task 497	Task 498	Task 499	Task 500	Task 501	Task 502	Task 503	Task 504	Task 505	Task 506	Task 507	Task 508	Task 509	Task 510	Task 511	Task 512	Task 513	Task 514	Task 515	Task 516	Task 517	Task 518	Task 519	Task 520	Task 521	Task 522	Task 523	Task 524	Task 525	Task 526	Task 527	Task 528	Task 529	Task 530	Task 531	Task 532	Task 533	Task 534	Task 535	Task 536	Task 537	Task 538	Task 539	Task 540	Task 541	Task 542	Task 543	Task 544	Task 545	Task 546	Task 547	Task 548	Task 549	Task 550	Task 551	Task 552	Task 553	Task 554	Task 555	Task 556	Task 557	Task 558	Task 559	Task 560	Task 561	Task 562	Task 563	Task 564	Task 565	Task 566	Task 567	Task 568	Task 569	Task 570	Task 571	Task 572	Task 573	Task 574	Task 575	Task 576	Task 577	Task 578	Task 579	Task 580	Task 581	Task 582	Task 583	Task 584	Task 585	Task 586	Task 587	Task 588	Task 589	Task 590	Task 591	Task 592	Task 593	Task 594	Task 595	Task 596	Task 597	Task 598	Task 599	Task 600	Task 601	Task 602	Task 603	Task 604	Task 605	Task 606	Task 607	Task 608	Task 609	Task 610	Task 611	Task 612	Task 613	Task 614	Task 615	Task 616	Task 617	Task 618	Task 619	Task 620	Task 621	Task 622	Task 623	Task 624	Task 625	Task 626	Task 627	Task 628	Task 629	Task 630	Task 631	Task 632	Task 633	Task 634	Task 635	Task 636	Task 637	Task 638	Task 639	Task 640	Task 641	Task 642	Task 643	Task 644	Task 645	Task 646	Task 647	Task 648	Task 649	Task 650	Task 651	Task 652	Task 653	Task 654	Task 655	Task 656	Task 657	Task 658	Task 659	Task 660	Task 661	Task 662	Task 663	Task 664	Task 665	Task 666	Task 667	Task 668	Task 669	Task 670	Task 671	Task 672	Task 673	Task 674	Task 675	Task 676	Task 677	Task 678	Task 679	Task 680	Task 681	Task 682	Task 683	Task 684	Task 685	Task 686	Task 687	Task 688	Task 689	Task 690	Task 691	Task 692	Task 693	Task 694	Task 695	Task 696	Task 697	Task 698	Task 699	Task 700	Task 701	Task 702	Task 703	Task 704	Task 705	Task 706	Task 707	Task 708	Task 709	Task 710	Task 711	Task 712	Task 713	Task 714	Task 715	Task 716	Task 717	Task 718	Task 719	Task 720	Task 721	Task 722	Task 723	Task 724	Task 725	Task 726	Task 727	Task 728	Task 729	Task 730	Task 731	Task 732	Task 733	Task 734	Task 735	Task 736	Task 737	Task 738	Task 739	Task 740	Task 741	Task 742	Task 743	Task 744	Task 745	Task 746	Task 747	Task 748	Task 749	Task 750	Task 751	Task 752	Task 753	Task 754	Task 755	Task 756	Task 757	Task 758	Task 759	Task 760	Task 761	Task 762	Task 763	Task 764	Task 765	Task 766	Task 767	Task 768	Task 769	Task 770	Task 771	Task 772	Task 773	Task 774	Task 775	Task 776	Task 777	Task 778	Task 779	Task 780	Task 781	Task 782	Task 783	Task 784	Task 785	Task 786	Task 787	Task 788	Task 789	Task 790	Task 791	Task 792	Task 793	Task 794	Task 795	Task 796	Task 797	Task 798	Task 799	Task 800	Task 801	Task 802	Task 803	Task 804	Task 805	Task 806	Task 807	Task 808	Task 809	Task 810	Task 811	Task 812	Task 813	Task 814	Task 815	Task 816	Task 817	Task 818	Task 819	Task 820	Task 821	Task 822	Task 823	Task 824	Task 825	Task 826	Task 827	Task 828	Task 829	Task 830	Task 831	Task 832	Task 833	Task 834	Task 835	Task 836	Task 837	Task 838	Task 839	Task 840	Task 841	Task 842
--------	--------	--------	--------	--------	--------	--------	--------	--------	--------	---------	---------	---------	---------	---------	---------	---------	---------	---------	---------	---------	---------	---------	---------	---------	---------	---------	---------	---------	---------	---------	---------	---------	---------	---------	---------	---------	---------	---------	---------	---------	---------	---------	---------	---------	---------	---------	---------	---------	---------	---------	---------	---------	---------	---------	---------	---------	---------	---------	---------	---------	---------	---------	---------	---------	---------	---------	---------	---------	---------	---------	---------	---------	---------	---------	---------	---------	---------	---------	---------	---------	---------	---------	---------	---------	---------	---------	---------	---------	---------	---------	---------	---------	---------	---------	---------	---------	---------	---------	---------	----------	----------	----------	----------	----------	----------	----------	----------	----------	----------	----------	----------	----------	----------	----------	----------	----------	----------	----------	----------	----------	----------	----------	----------	----------	----------	----------	----------	----------	----------	----------	----------	----------	----------	----------	----------	----------	----------	----------	----------	----------	----------	----------	----------	----------	----------	----------	----------	----------	----------	----------	----------	----------	----------	----------	----------	----------	----------	----------	----------	----------	----------	----------	----------	----------	----------	----------	----------	----------	----------	----------	----------	----------	----------	----------	----------	----------	----------	----------	----------	----------	----------	----------	----------	----------	----------	----------	----------	----------	----------	----------	----------	----------	----------	----------	----------	----------	----------	----------	----------	----------	----------	----------	----------	----------	----------	----------	----------	----------	----------	----------	----------	----------	----------	----------	----------	----------	----------	----------	----------	----------	----------	----------	----------	----------	----------	----------	----------	----------	----------	----------	----------	----------	----------	----------	----------	----------	----------	----------	----------	----------	----------	----------	----------	----------	----------	----------	----------	----------	----------	----------	----------	----------	----------	----------	----------	----------	----------	----------	----------	----------	----------	----------	----------	----------	----------	----------	----------	----------	----------	----------	----------	----------	----------	----------	----------	----------	----------	----------	----------	----------	----------	----------	----------	----------	----------	----------	----------	----------	----------	----------	----------	----------	----------	----------	----------	----------	----------	----------	----------	----------	----------	----------	----------	----------	----------	----------	----------	----------	----------	----------	----------	----------	----------	----------	----------	----------	----------	----------	----------	----------	----------	----------	----------	----------	----------	----------	----------	----------	----------	----------	----------	----------	----------	----------	----------	----------	----------	----------	----------	----------	----------	----------	----------	----------	----------	----------	----------	----------	----------	----------	----------	----------	----------	----------	----------	----------	----------	----------	----------	----------	----------	----------	----------	----------	----------	----------	----------	----------	----------	----------	----------	----------	----------	----------	----------	----------	----------	----------	----------	----------	----------	----------	----------	----------	----------	----------	----------	----------	----------	----------	----------	----------	----------	----------	----------	----------	----------	----------	----------	----------	----------	----------	----------	----------	----------	----------	----------	----------	----------	----------	----------	----------	----------	----------	----------	----------	----------	----------	----------	----------	----------	----------	----------	----------	----------	----------	----------	----------	----------	----------	----------	----------	----------	----------	----------	----------	----------	----------	----------	----------	----------	----------	----------	----------	----------	----------	----------	----------	----------	----------	----------	----------	----------	----------	----------	----------	----------	----------	----------	----------	----------	----------	----------	----------	----------	----------	----------	----------	----------	----------	----------	----------	----------	----------	----------	----------	----------	----------	----------	----------	----------	----------	----------	----------	----------	----------	----------	----------	----------	----------	----------	----------	----------	----------	----------	----------	----------	----------	----------	----------	----------	----------	----------	----------	----------	----------	----------	----------	----------	----------	----------	----------	----------	----------	----------	----------	----------	----------	----------	----------	----------	----------	----------	----------	----------	----------	----------	----------	----------	----------	----------	----------	----------	----------	----------	----------	----------	----------	----------	----------	----------	----------	----------	----------	----------	----------	----------	----------	----------	----------	----------	----------	----------	----------	----------	----------	----------	----------	----------	----------	----------	----------	----------	----------	----------	----------	----------	----------	----------	----------	----------	----------	----------	----------	----------	----------	----------	----------	----------	----------	----------	----------	----------	----------	----------	----------	----------	----------	----------	----------	----------	----------	----------	----------	----------	----------	----------	----------	----------	----------	----------	----------	----------	----------	----------	----------	----------	----------	----------	----------	----------	----------	----------	----------	----------	----------	----------	----------	----------	----------	----------	----------	----------	----------	----------	----------	----------	----------	----------	----------	----------	----------	----------	----------	----------	----------	----------	----------	----------	----------	----------	----------	----------	----------	----------	----------	----------	----------	----------	----------	----------	----------	----------	----------	----------	----------	----------	----------	----------	----------	----------	----------	----------	----------	----------	----------	----------	----------	----------	----------	----------	----------	----------	----------	----------	----------	----------	----------	----------	----------	----------	----------	----------	----------	----------	----------	----------	----------	----------	----------	----------	----------	----------	----------	----------	----------	----------	----------	----------	----------	----------	----------	----------	----------	----------	----------	----------	----------	----------	----------	----------	----------	----------	----------	----------	----------	----------	----------	----------	----------	----------	----------	----------	----------	----------	----------	----------	----------	----------	----------	----------	----------	----------	----------	----------	----------	----------	----------	----------	----------	----------	----------	----------	----------	----------	----------	----------	----------	----------	----------	----------	----------	----------	----------	----------	----------	----------	----------	----------	----------	----------	----------	----------	----------	----------	----------	----------	----------	----------	----------	----------	----------	----------	----------	----------	----------	----------	----------	----------	----------	----------	----------	----------	----------	----------	----------	----------	----------	----------	----------	----------	----------	----------	----------	----------	----------	----------	----------	----------	----------	----------	----------	----------	----------	----------	----------	----------	----------	----------	----------	----------	----------	----------	----------	----------	----------	----------	----------	----------	----------	----------	----------	----------	----------	----------	----------	----------	----------	----------	----------	----------	----------	----------	----------	----------	----------	----------	----------	----------	----------	----------	----------

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...

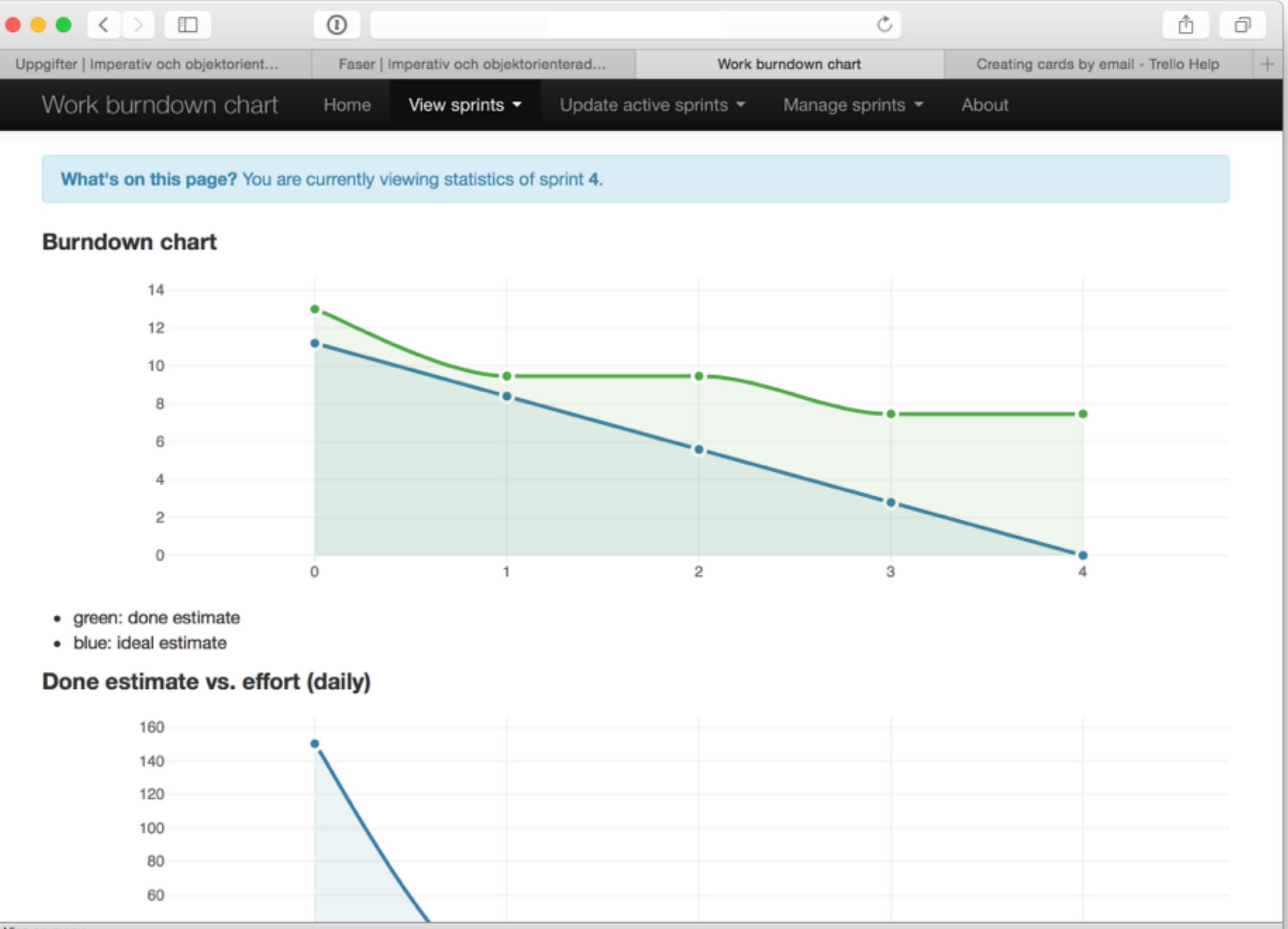
Add

- Members
- Labels
- Checklist
- Due Date
- Attachment

Actions

- Move
- Copy
- Subscribe
- Archive

[Share and more...](#)



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 7.2 and ending at 0. The actual velocity starts at 7.2, dips slightly, rises to a peak of about 6.2, and then decreases more sharply than the ideal line, reaching 0 by the end of the sprint.

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 beröra 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 skräpsamlare behöver inte programmeraren orsa 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 skräpsamlare 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 rörs.

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 skräpsamlare. Detta är särskilt bra av utgående med referensräkning eftersom det kräver en liten del extra arbete borta för att uppdatera objekters referensräknare.³ Men ju före en referens räknas eller ändras en enda objekts läge upp i minnet och dess referensräknare uppdateras. Det kan medföra stora misstag eftersom objekten tas upp i codeminnet, vilket påverkar positionen negativt.

Det kanske största problemet med referensräknande skräpsamlare 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 misstag extra arbete.

2.2 Mark and sweep

Mark and sweep tillhör instängen av tracing eller spårande skräpsamlare. 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 skräpsamlingen att vid olika tillfällen. Ofta när detta när många lagringar minne tas ut eller faller under en viss nivå.

Mark and sweep har två steg som inte heller kräver kallas för "mark" och "sweep". Första steget mark, gör ut på att hitta och markera alla levande objekt. Själva markeringen sätter genast ett flagga som sparar tillsammans med varje objekts sätta. Ista "mark" steget påbörjas nödigt samtidigt flaggor.

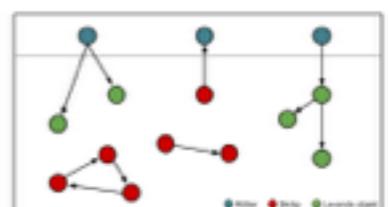


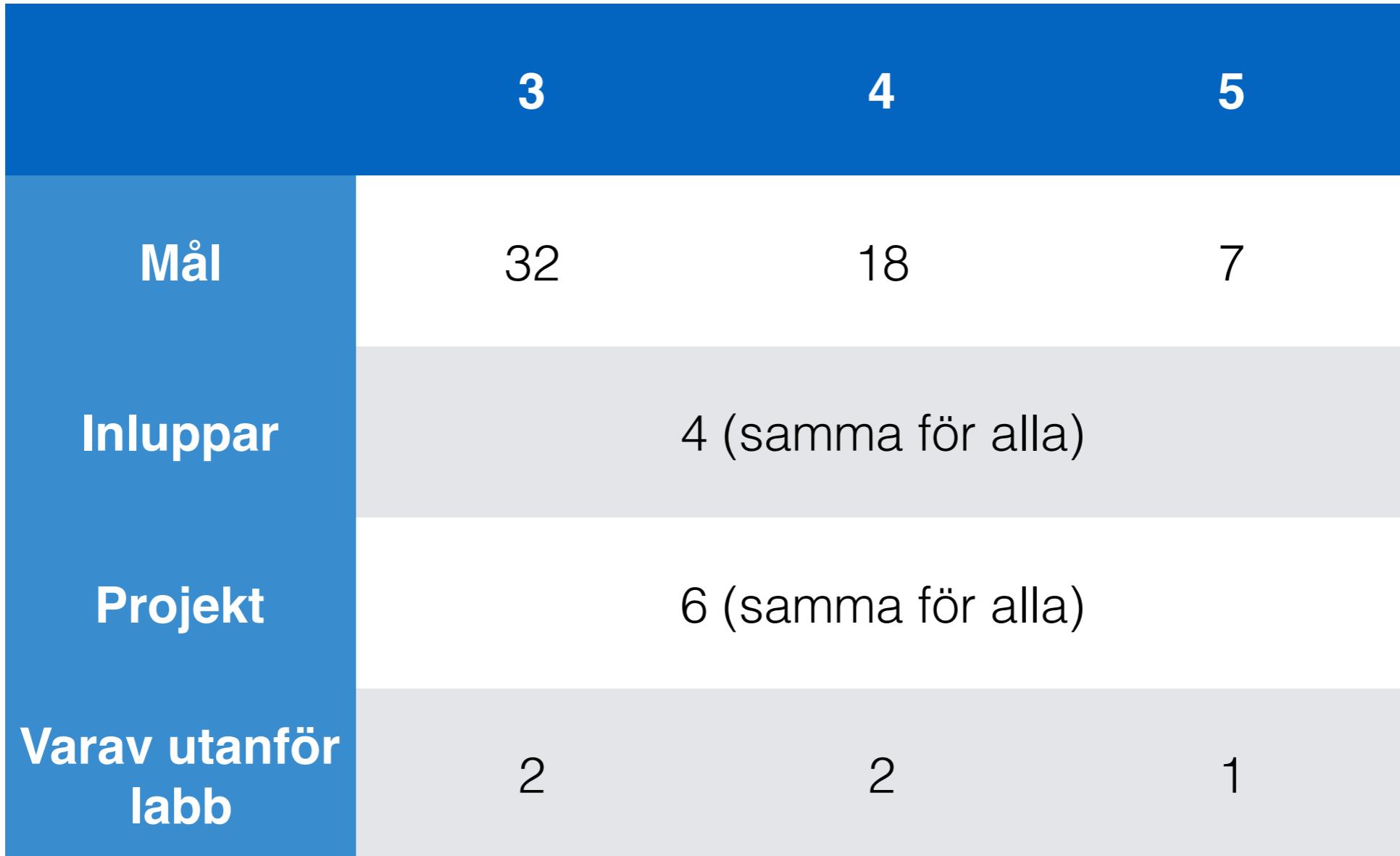
Figure 1: Objekter representerar objekt och pilarna referenser.

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 tifjar dessa referenser. Varje objekt som hittas markeras och för objekts senliga referenser är upprepa minsta process som för rötterna. Alltså vi tifjar referenser och alla objekt som hittas markeras och dessa referenser flyttas. 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", hämtat 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**

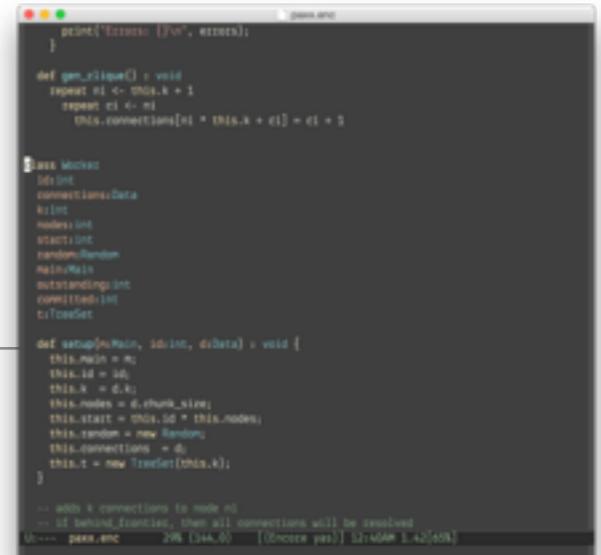
Gedit

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**