

Projekt i TDDDD78 / TDDE30

...Utforska, skapa ett eget projekt!

- I projektet ska ni:
 - Designa och implementera ett OO-program
 - Utan detaljerad styrning från en fördefinierad uppgift
 - Visa upp resultatet – *vid demo/inlämning*
 - Få programkoden granskad med feedback om eventuella problem
 - Kanske komplettera efter granskningen...

- Projekt väljs fritt, men:
 - **Inte** fortsättning på Tetris
 - **Inte** för likt Tetris (riskabelt med Breakout, Snake, ...)
 - **Inte** Android-projekt (för mycket Android-specifika ramverk)
- Visa vad ni har lärt er om objektorienterad programmering
 - **Undvik** projekt som kräver alltför mycket:
 - "Lågnivåprogrammering", matrismatematik, ...
 - Komplex grafik, ljud, animering
 - ...

Exempel på tidigare projekt

Exempel

Sonus





Sweet Tunes

SAVE

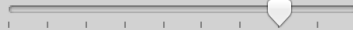
LOAD

DELETE

EXPORT

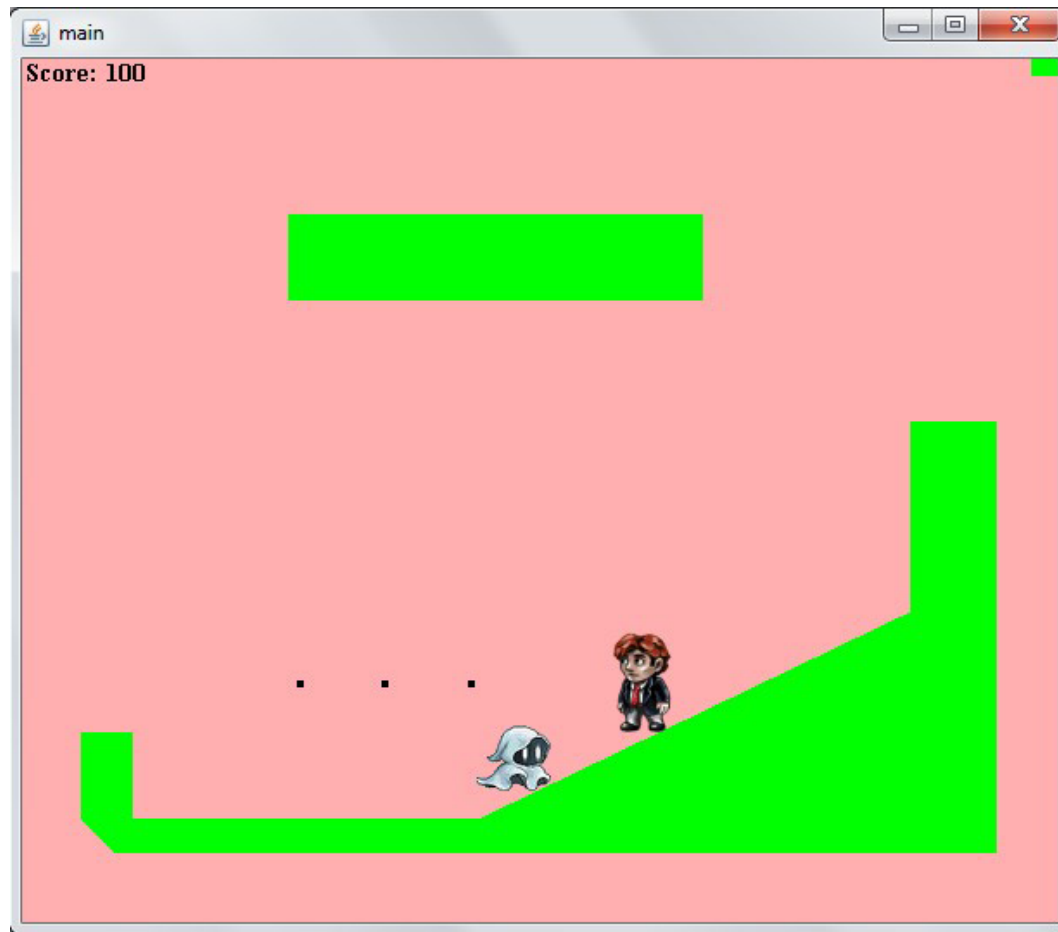




Tempo 

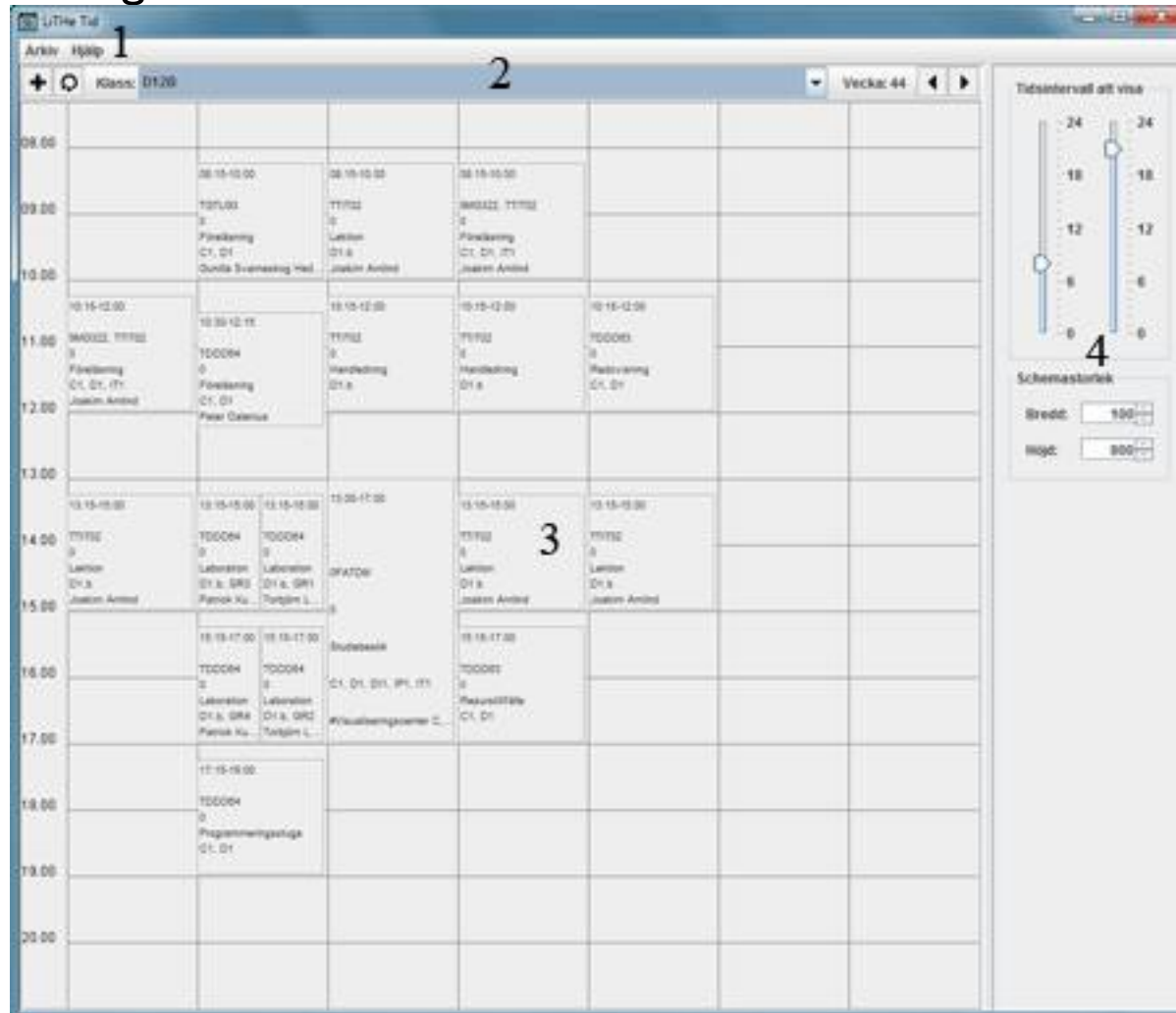
Exempel

- Sidoscrollande spel



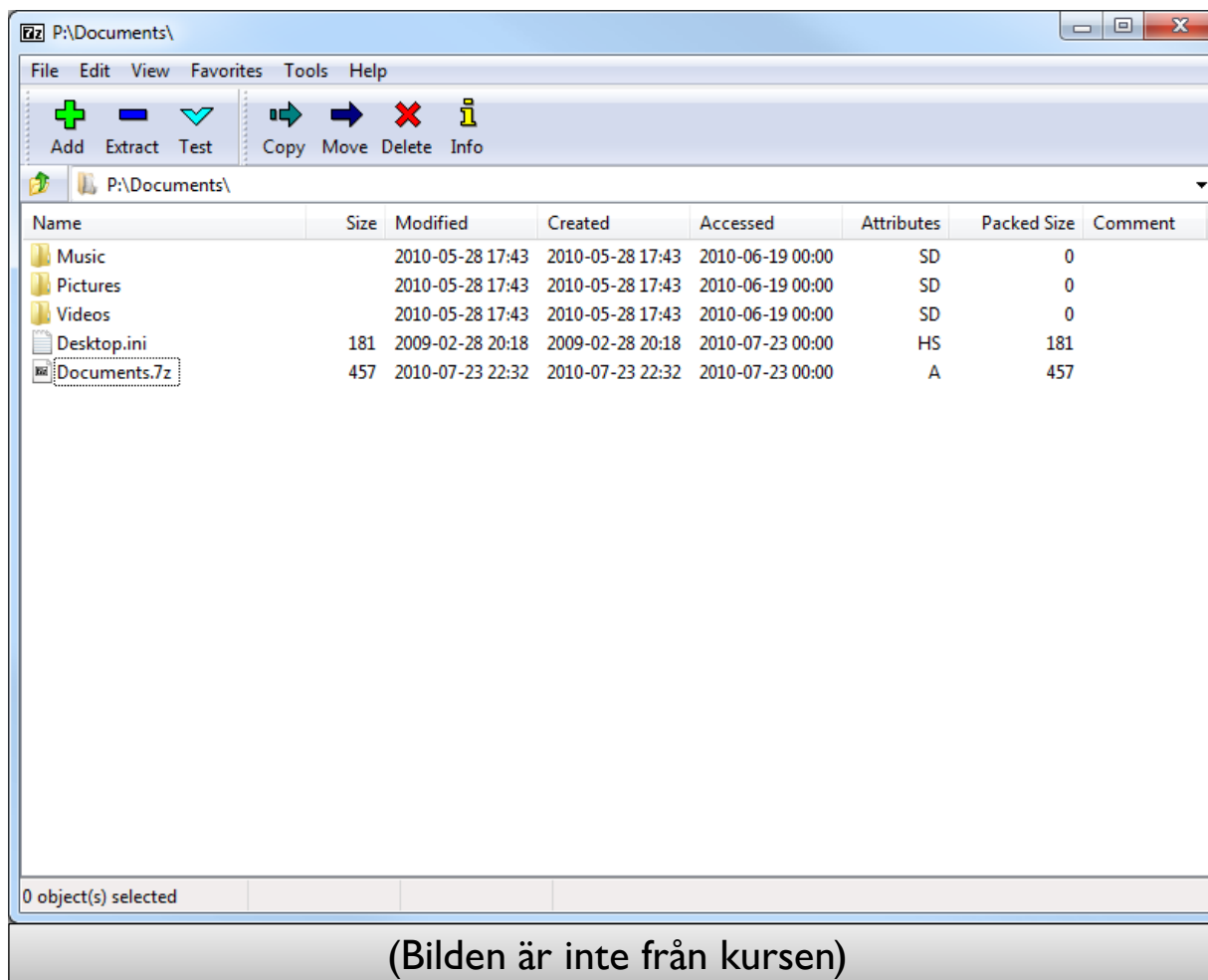
Exempel

- Kalenderprogram
 - Kanske med gränssnitt mot TimeEdit



[illegible]

■ Zip-verktyg

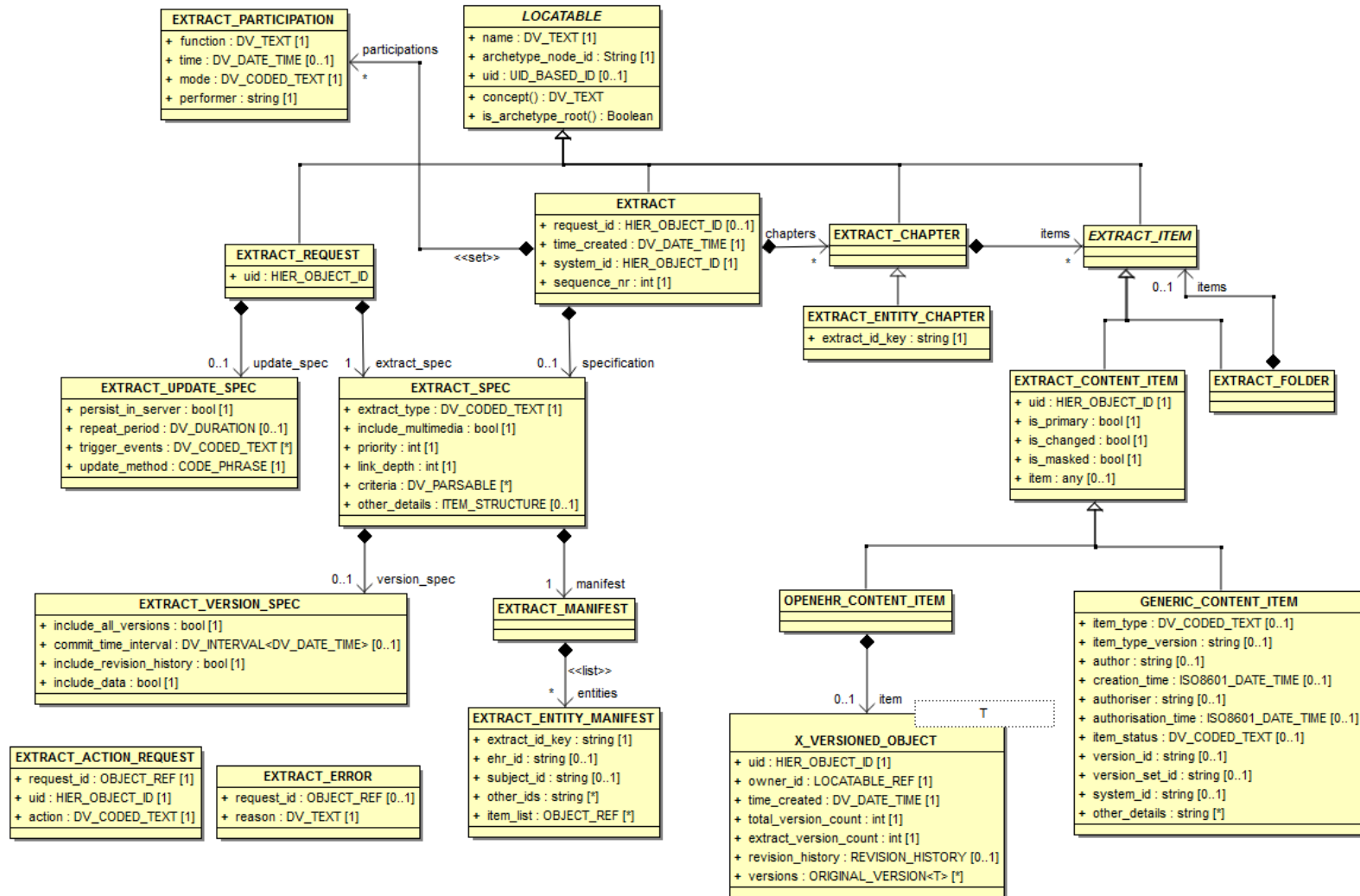


Exempel

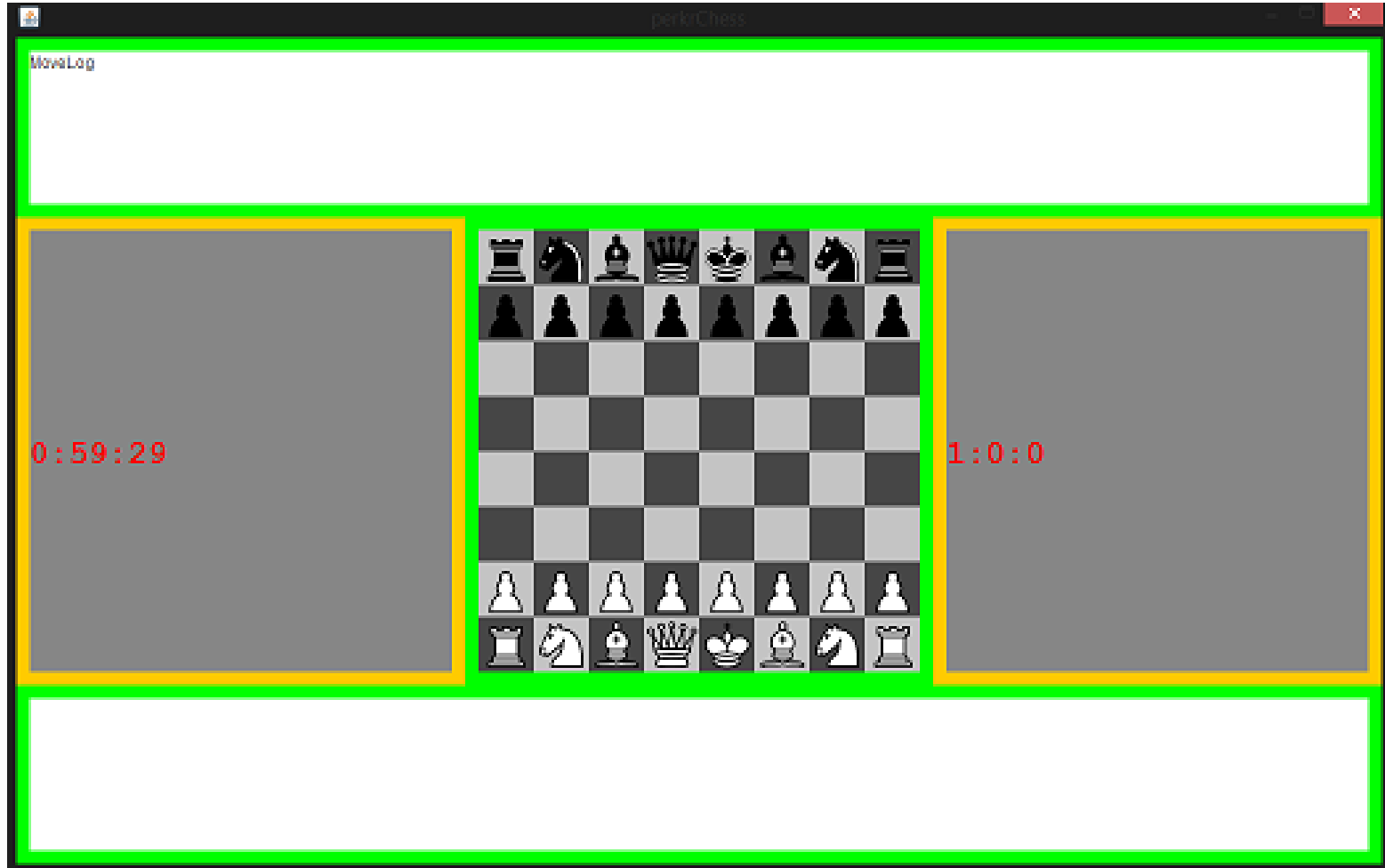
- Treasure Hunter



UML-visare

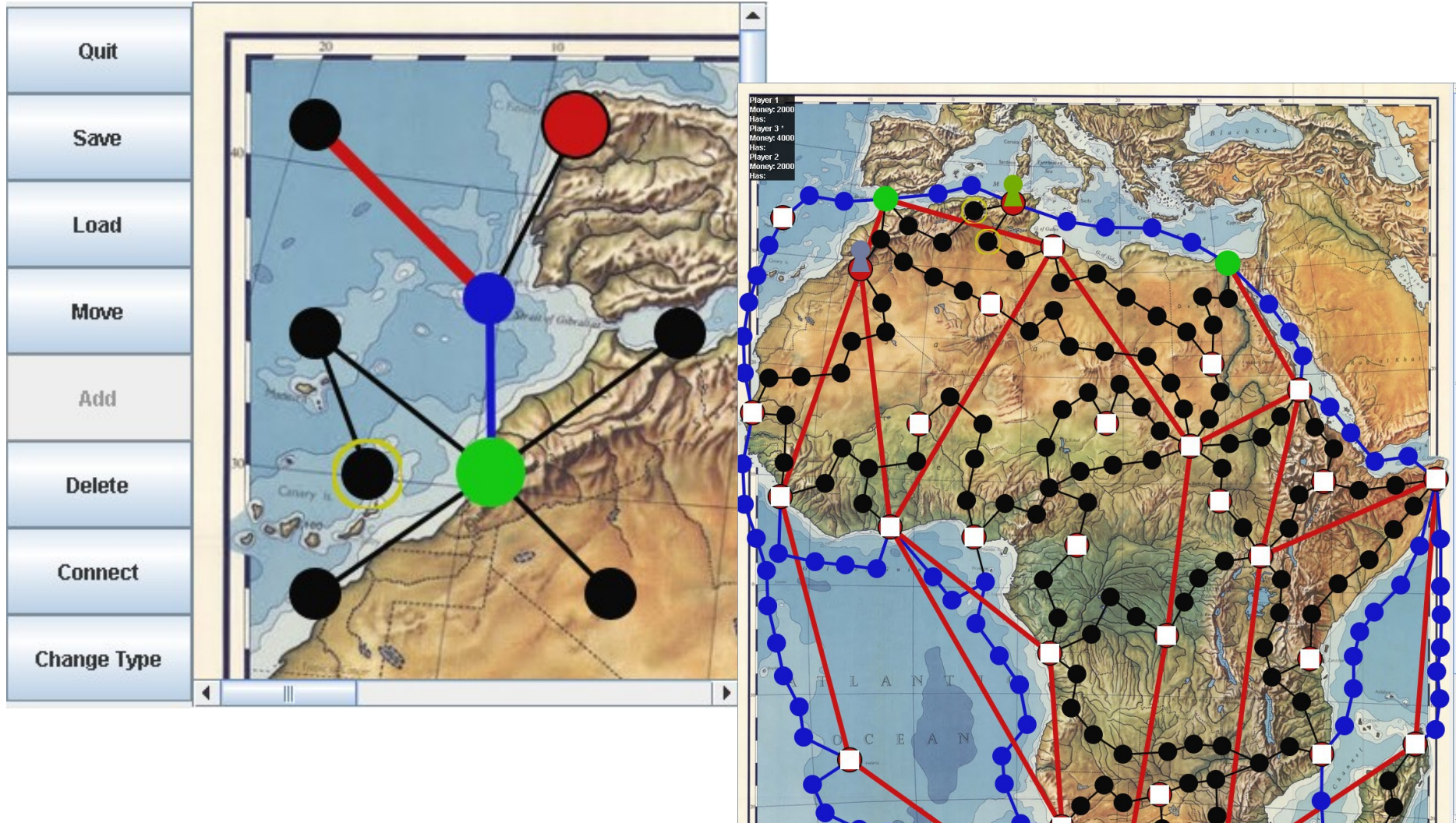


- Schack

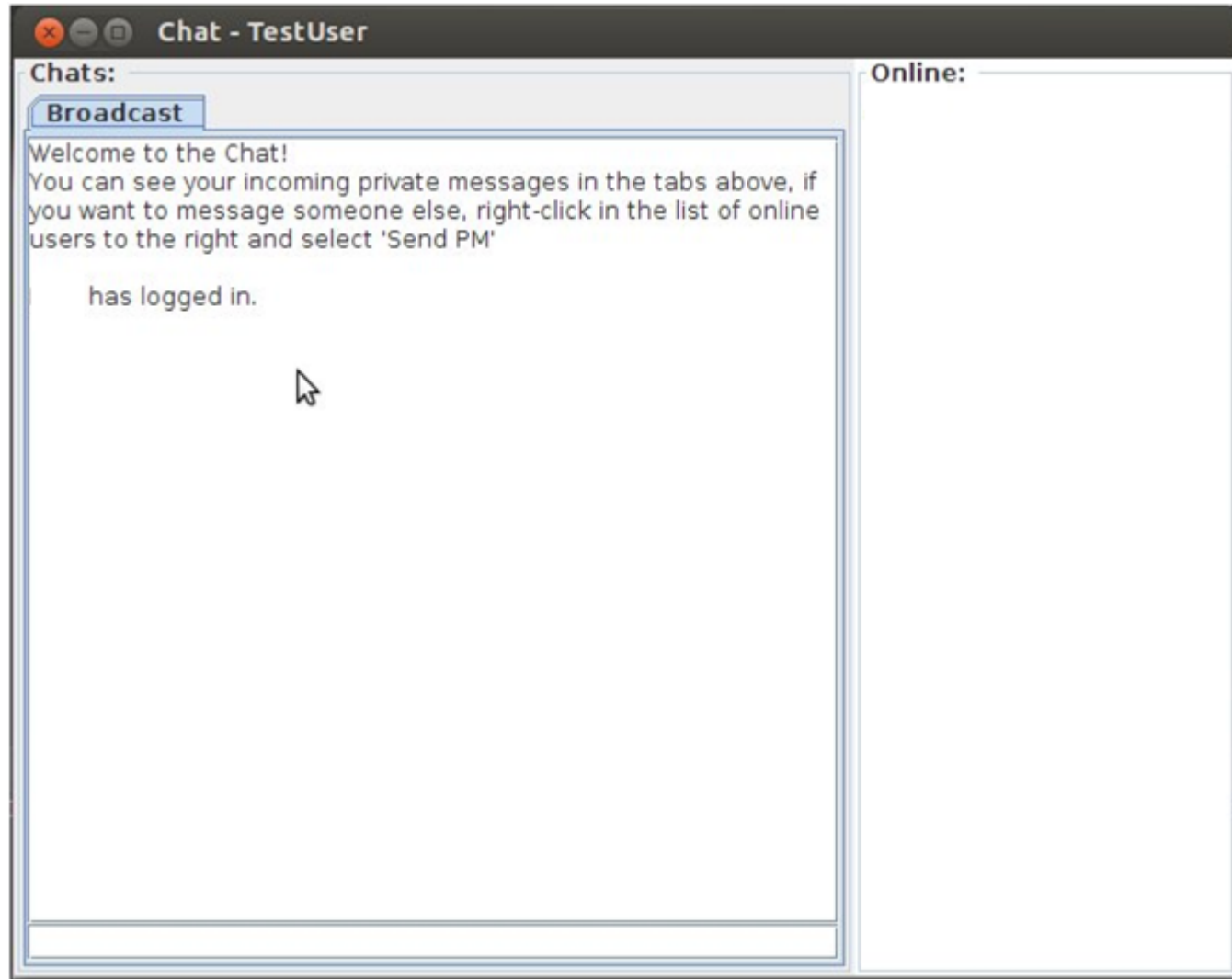


Exempel

■ Brädspel



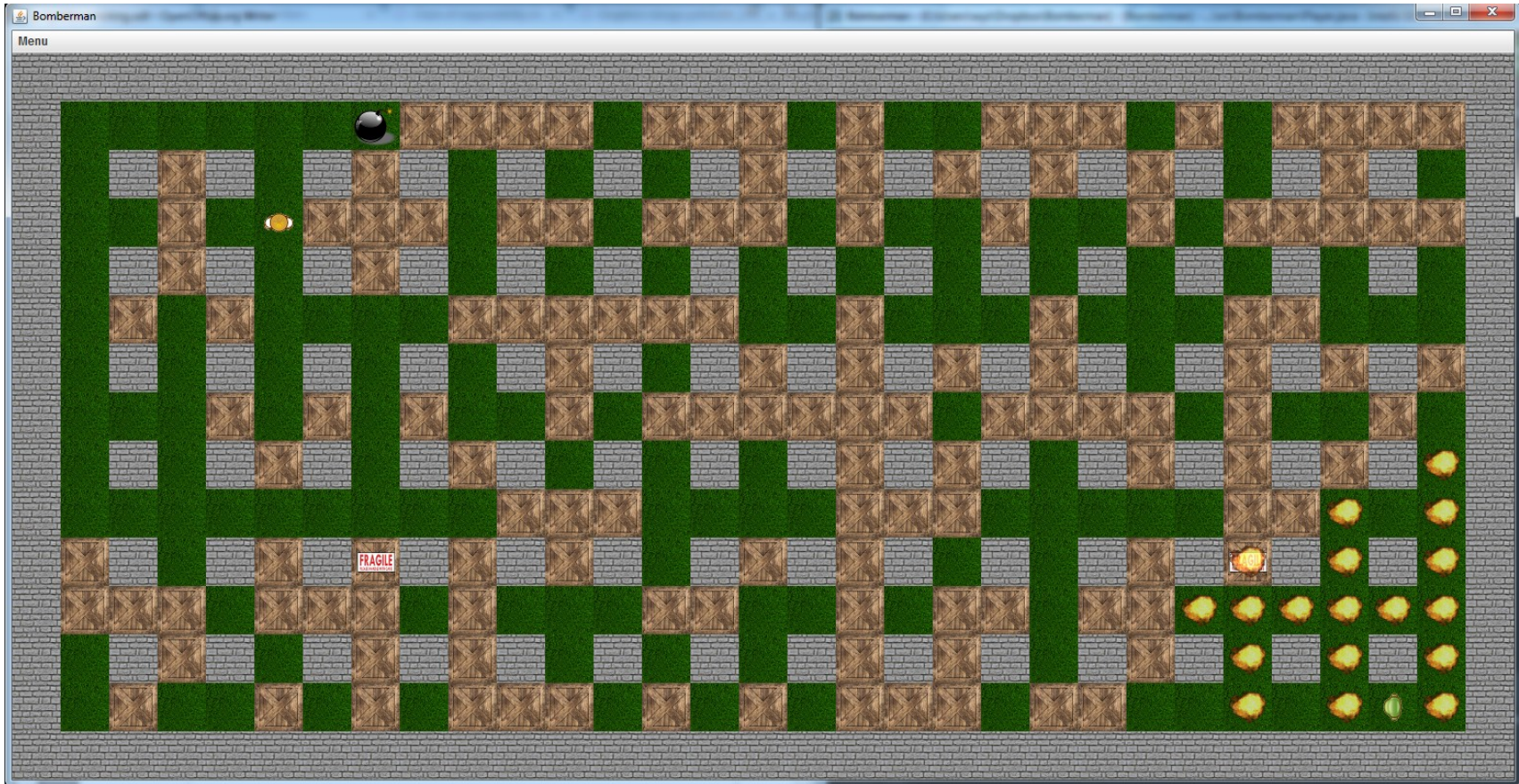
■ Chat-system



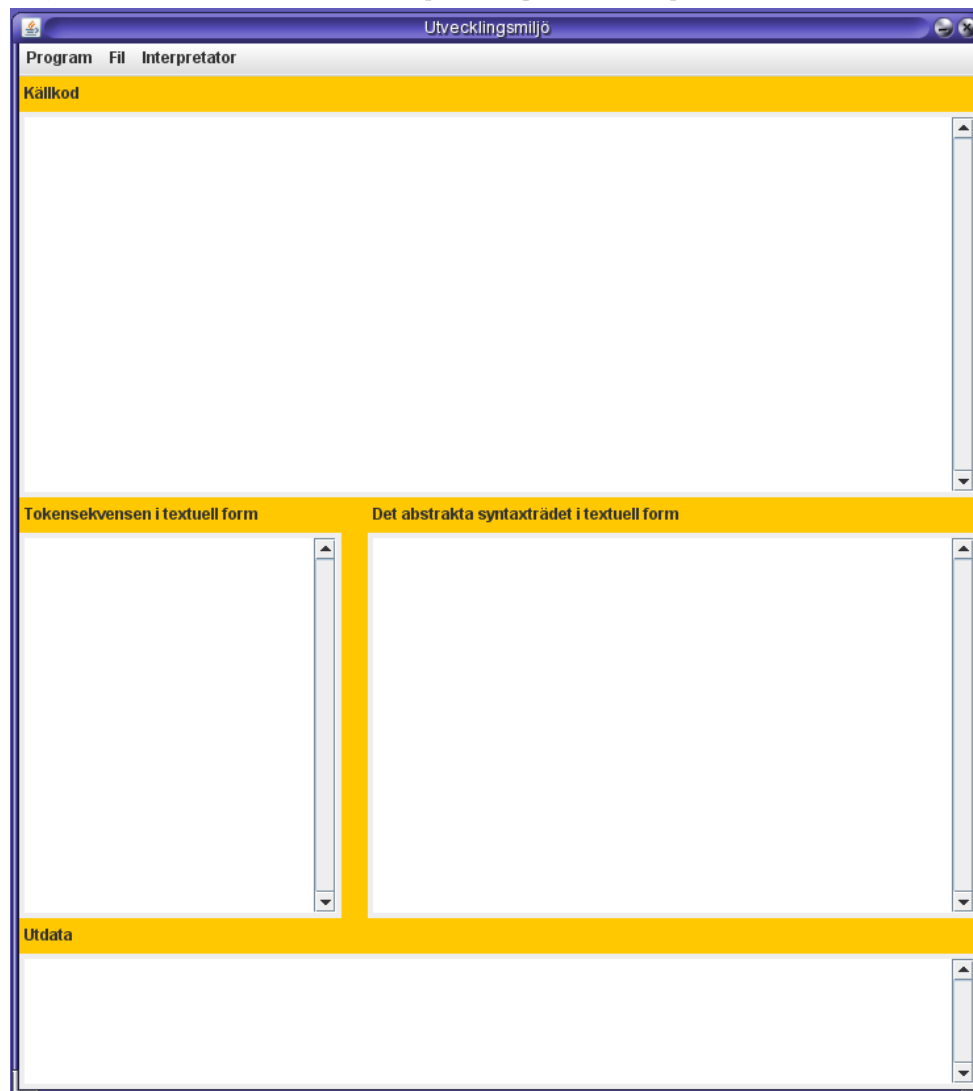
- Spel



- Spel



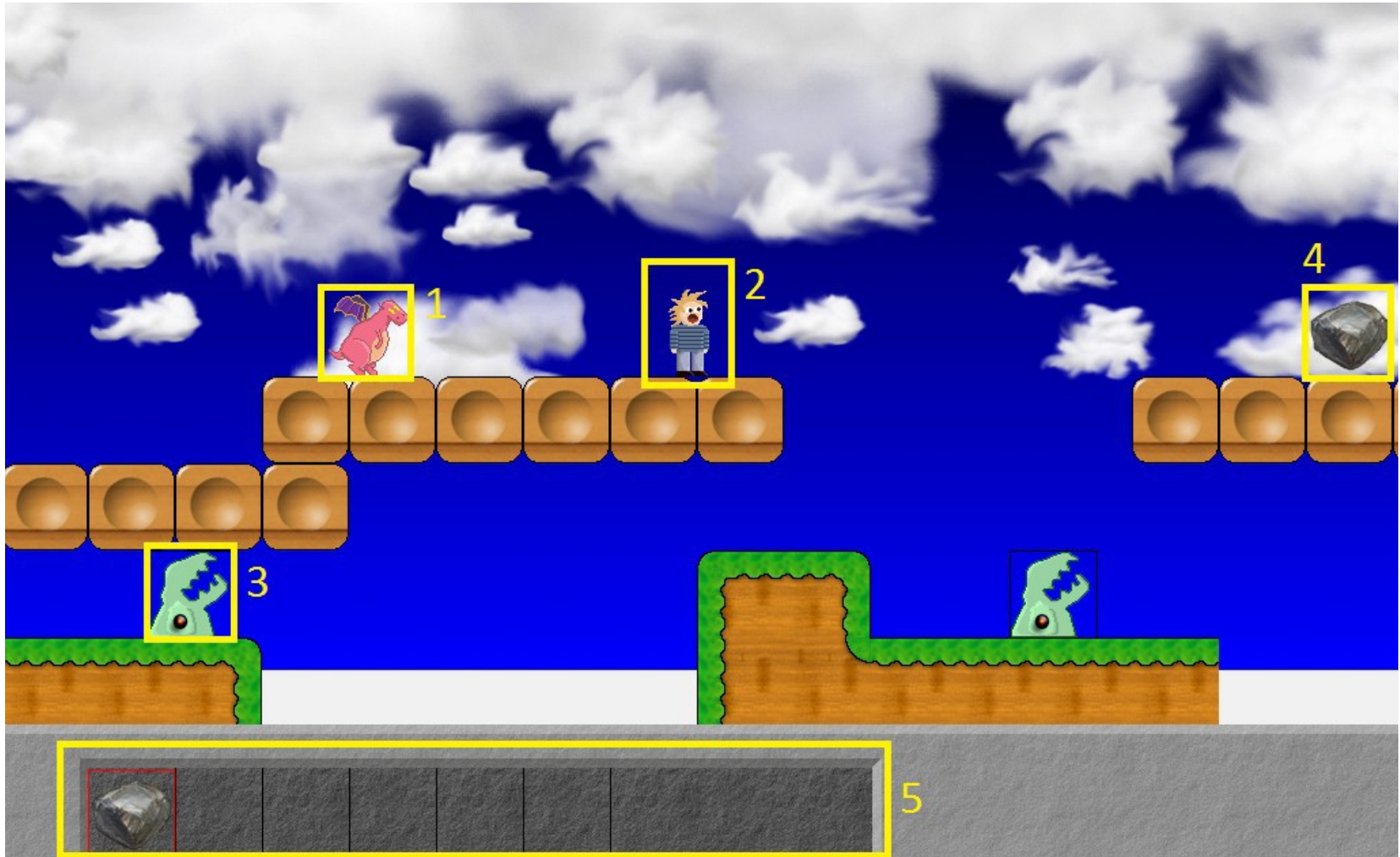
- Utvecklingsmiljö för ett litet programspråk



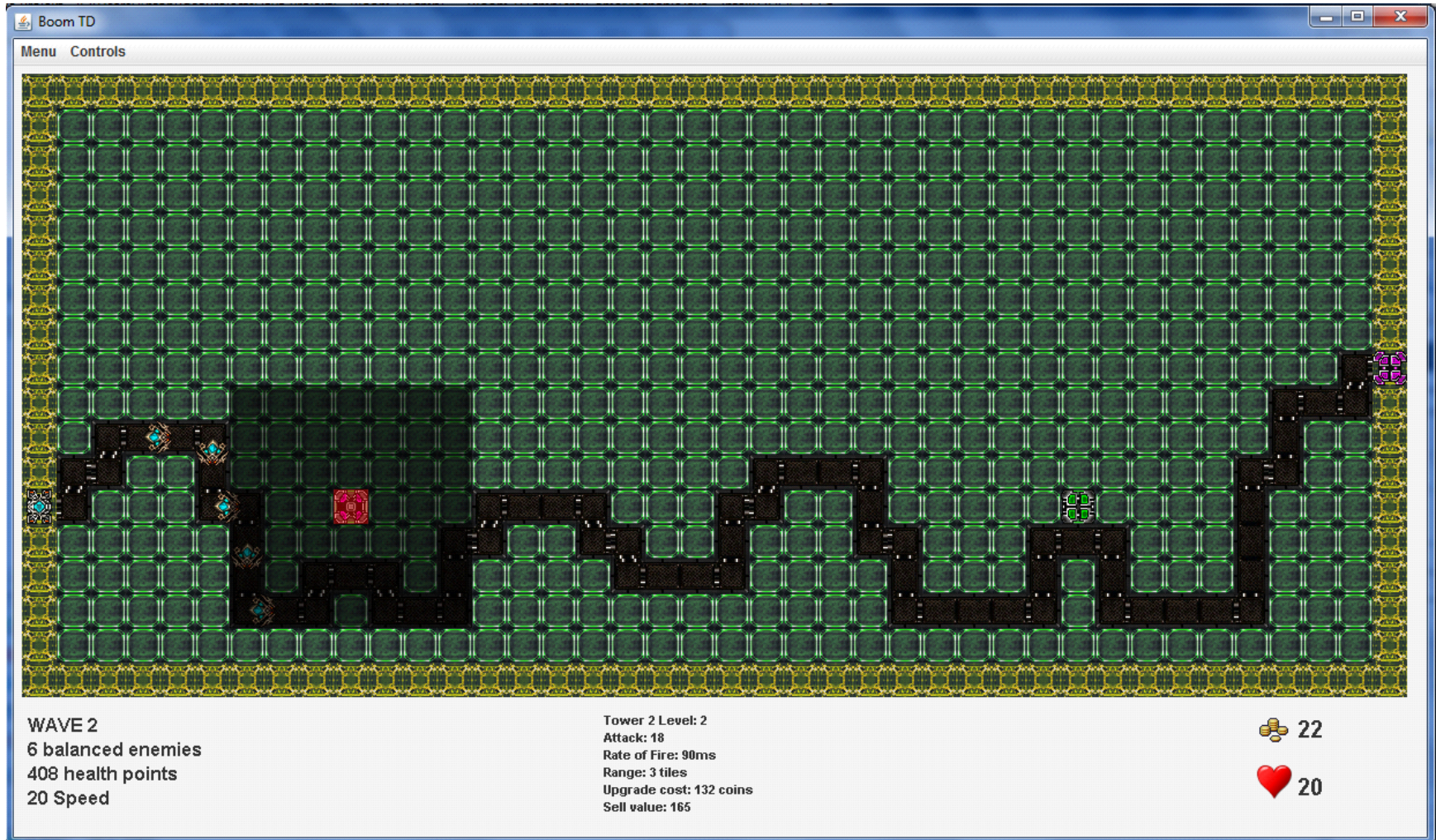
- Bilspel



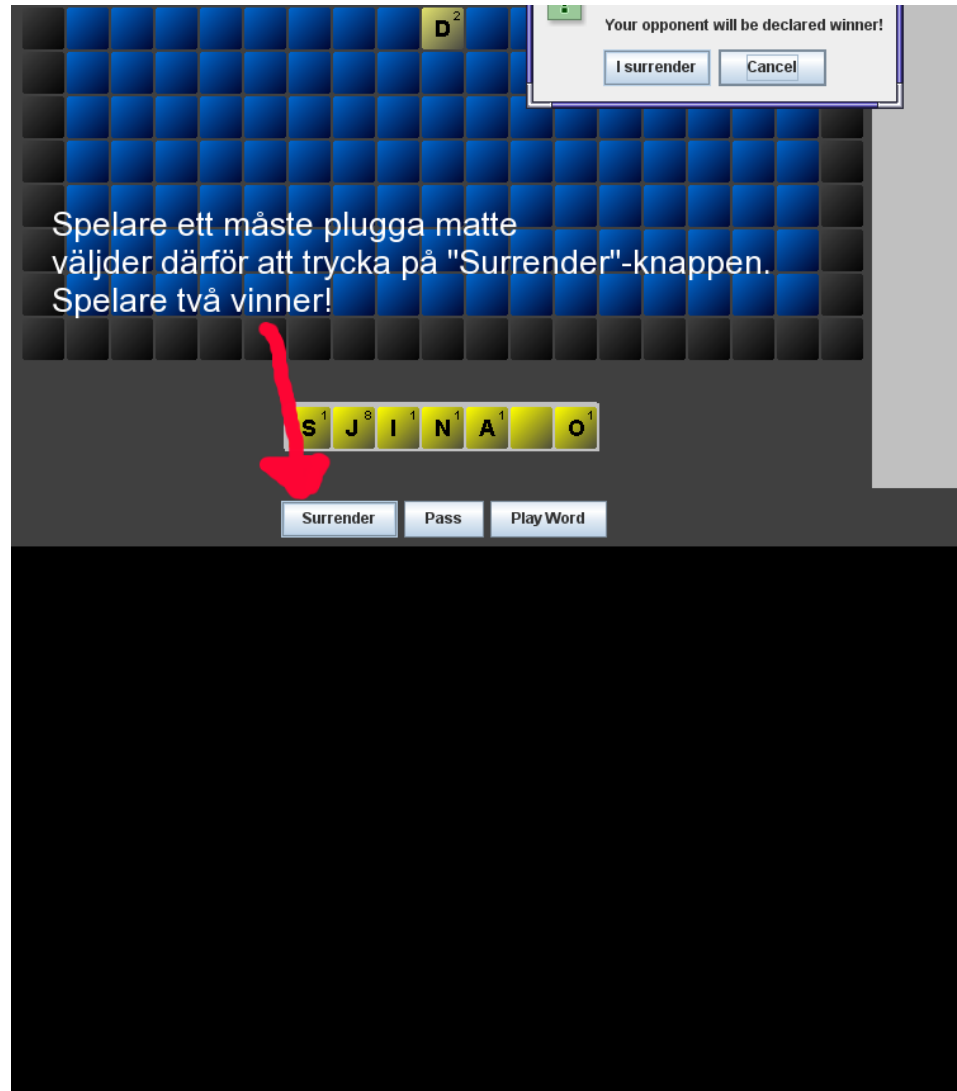
- Plattformsspel



■ Tower Defense

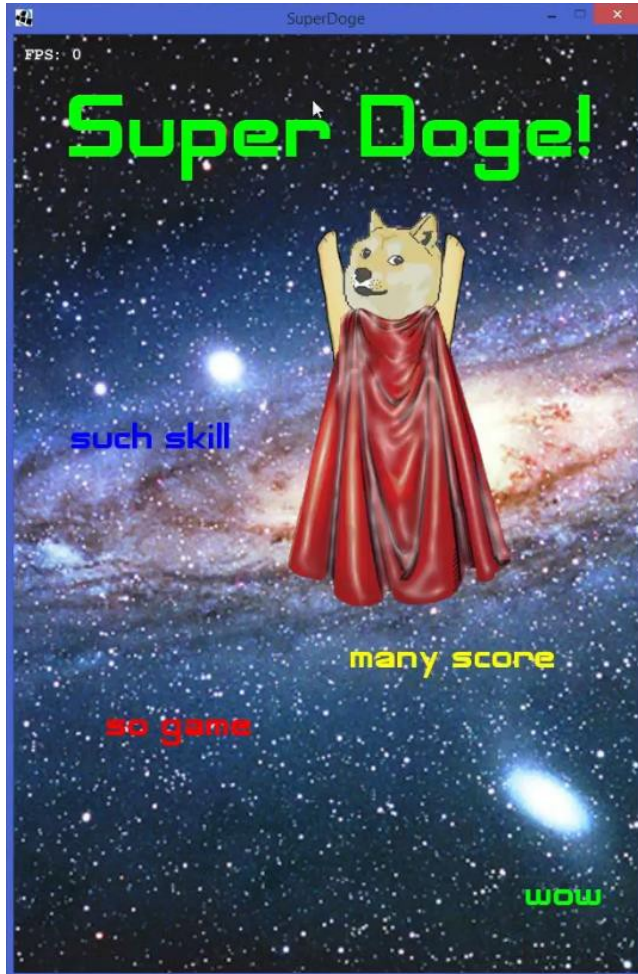


■ Word Wars



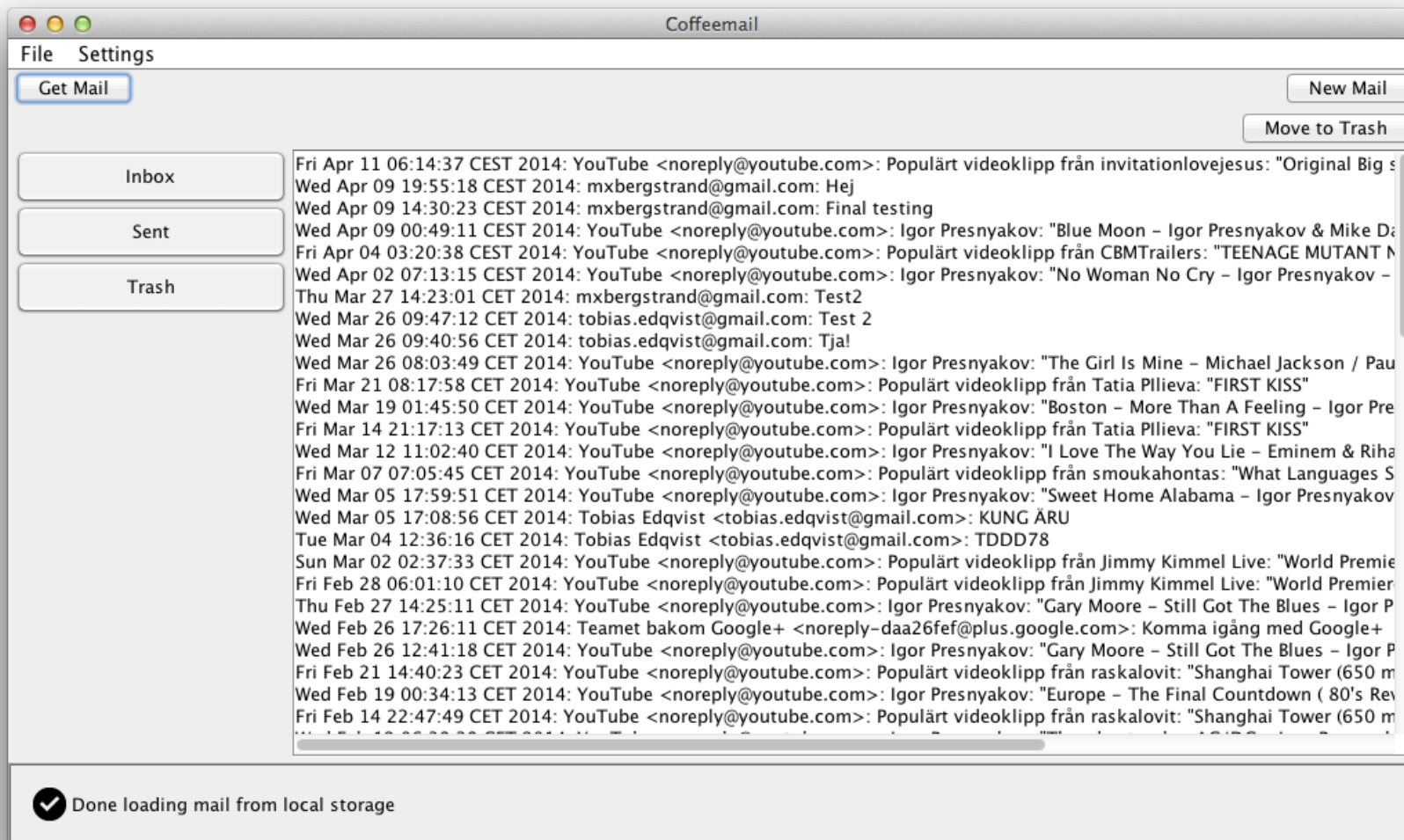
Exempel

- Sidoscrollande spel

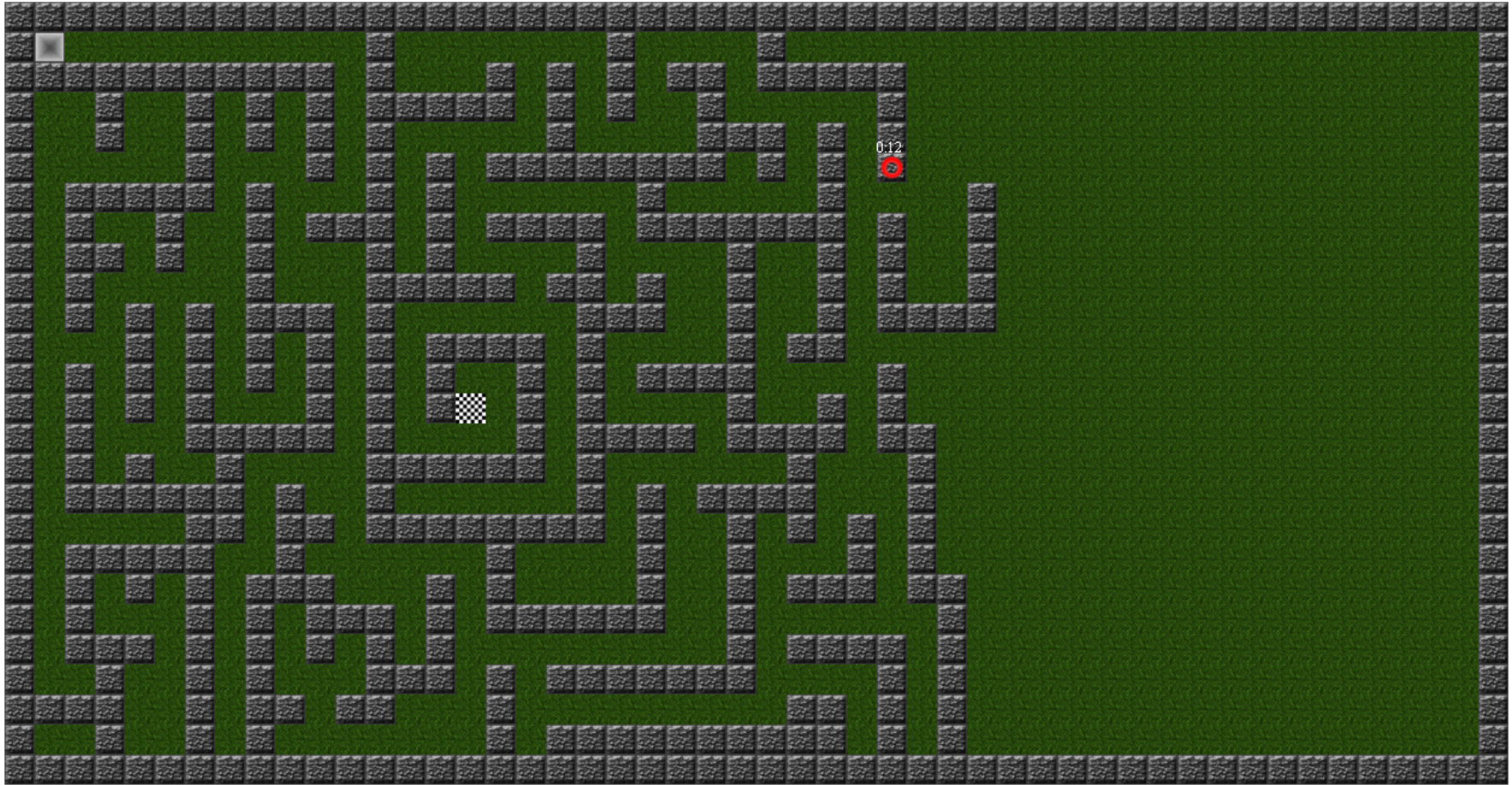


Exempel





Exempel



Exempel

All shows

The screenshot shows a web application for managing TV shows. It features a 'File' menu at the top left, followed by filters for 'Air date', 'Unwatched', and 'Delete show'. On the right, there are buttons for 'Mark as: Watched', 'Unwatched', 'Update', and 'Add show'.

The main content area is divided into three sections:

- All shows:** A list of TV shows with their air dates. 'Continuum' is highlighted in blue.
- Episodes of selected show:** A list of episodes for the selected show, 'Continuum'. The episode 'Minute To Win It' (S03E03) is highlighted in blue.
- Details of selected episode:** A detailed view of the selected episode, 'Minute To Win It' (S03E03), including a synopsis and a poster image.

Annotations with arrows point to these sections:

- A black arrow points from the text 'All shows' to the first section.
- A pink arrow points from the text 'Episodes of selected show' to the second section.
- A green arrow points from the text 'Details of selected episode' to the third section.

Show	Air date
Arrow	2014-04-02
The Big Bang Theory	2014-04-03
Hannibal	2014-04-04
Hat of Dixie	2014-04-04
Continuum	2014-04-06
The Blacklist	2014-04-07
The Following	2014-04-07
The Mentalist	2014-04-13
Person of Interest	2014-04-15
House of Cards (US)	Unknown
True Detective	Unknown

Episode	Air date
Minute By Minute	S03E01
Minute Man	S03E02
Minute To Win It	S03E03
A Minute Changes Everything	S03E04
30 Minutes To Air	S03E05
tba	S03E06
tba	S03E07
So Do Our Minutes Hasten	S03E08
tba	S03E09
tba	S03E10
tba	S03E11

Minute To Win It
S03E03
Mark as unwatched
2014-03-30

Kiera has to solve a series of Liber8 connected bank heists and discovers the robbers are controlled by a recently escaped Lucas. Carlos' new and unwanted knowledge about time travel begins to affect his work.

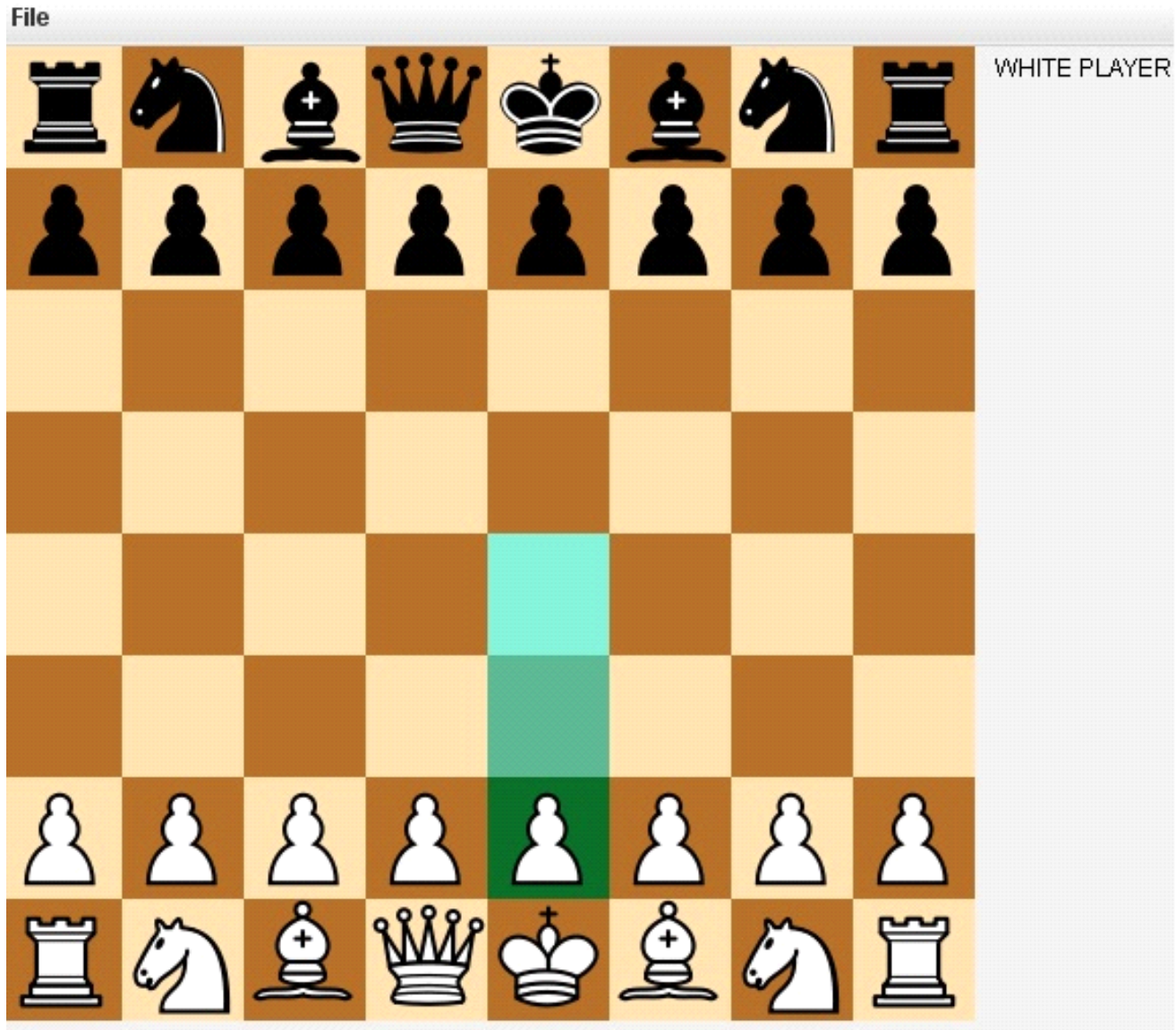


CONTINUUM

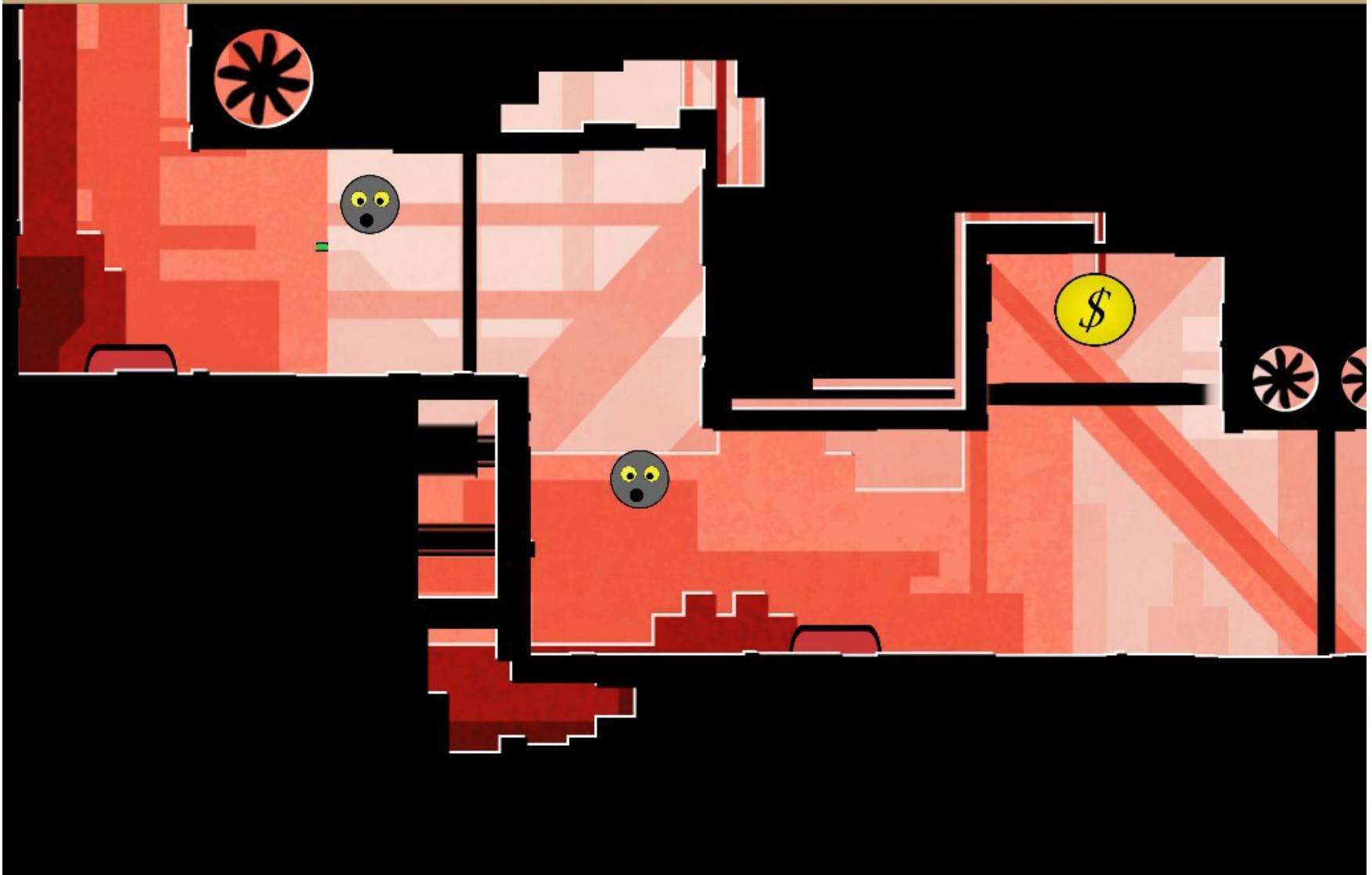
Episodes of selected show

Details of selected episode

Exempel



Exempel



Tillsammans eller ensam?

Allmänt: Arbeta gärna i par



Ny gruppindelning, ensam eller grupper om 2 –
cirka 80 timmar arbete per person

Kan samarbeta över klassgränser,
kanske schemakrockar över programgränser: [D1a/b/c] — [U1a/b]

Jag skulle också säga att valet att jobba själv nog ändå bör undvikas, även om det finns fördelar så tror jag att det är bättre att jobba i grupp. Att ha en projektpartner att diskutera med är lärorikt.

Att jobba själv har fördelar och nackdelar. En fördel är att det är lätt att hitta tider att jobba men samtidigt finns nackdelen att det kan vara svårt att faktiskt göra det då det inte finns någon annan som förväntar sig att man ska jobba.

- **Hitta projektpartner**, om du inte vill jobba ensam
 - På labbar och föreläsningar...
 - Hjälp: Kanalen "Söka projektpartner" i Java-2025-handuppräckning
 - Vi kan inte hitta en partner åt dig, och det är helt OK att arbeta ensam
- Kom överens om *projekt* och hur ni ska *arbeta*
- **Anmäl er i WebReg**
 - **D1**: Grupp 20-21-22
 - **U1**: Grupp 23-24 (anmälan öppnar snart)

**Att ta bort eller ändra anmälan ger extra arbete
(byta Gitlab-projekt, ändra Teams-kanaler, ...)**

Att planera projektet

D1

Före tentaperioden:

Tid avsatt till att *hitta på* ett projekt,
skriva *projektplan* – 250304-250312

Två perioder →
finns kalendertid att låta idéer mogna

Får baseras på inspirationsprojekten,
men gör gärna något eget

Inlämnat senast **250312** → vi läser,
kommenterar ev. *uppenbara problem*
innan projektstart 250331

Inlämnat efter detta →
vi granskar om/när vi får tid

U1

Inför projektstart:

Skriv en *projektplan*!

En period →
Mindre kalendertid för de egna idéerna

Inspirationsprojekt finns:
Outline med generella idéer att *bygga vidare* på, konkretisera, utveckla...

Inlämnat senast **250302** → vi läser,
kommenterar ev. *uppenbara problem*
så snart vi kan

Inlämnat efter detta →
vi granskar om/när vi får tid

Vi kommenterar bara uppenbara problem!

Inga garantier att vi hittar allt – eget ansvar...

Projektplan ska finnas!

Även om ni bara skulle lämna in den
som del av slutinlämningen...

- Planera en sekvens av **stegvisa utökningar!**
 - Liten "kärna" som kan implementeras snabbt
 - Många finesser som kan implementeras efter hand
 - Möjlighet att stoppa när tiden är slut!

- ➔ Lista på milstolpar i projektplanen:
 - 1. ...
 - 2. ...
 - 3. ...
 - ...
 - 14. ...
 - 15. ...
 - ...
 - ...
 - 35. ...

**Enkelt att implementera,
ändå testbart**

**Lagom stora steg
att utöka med**

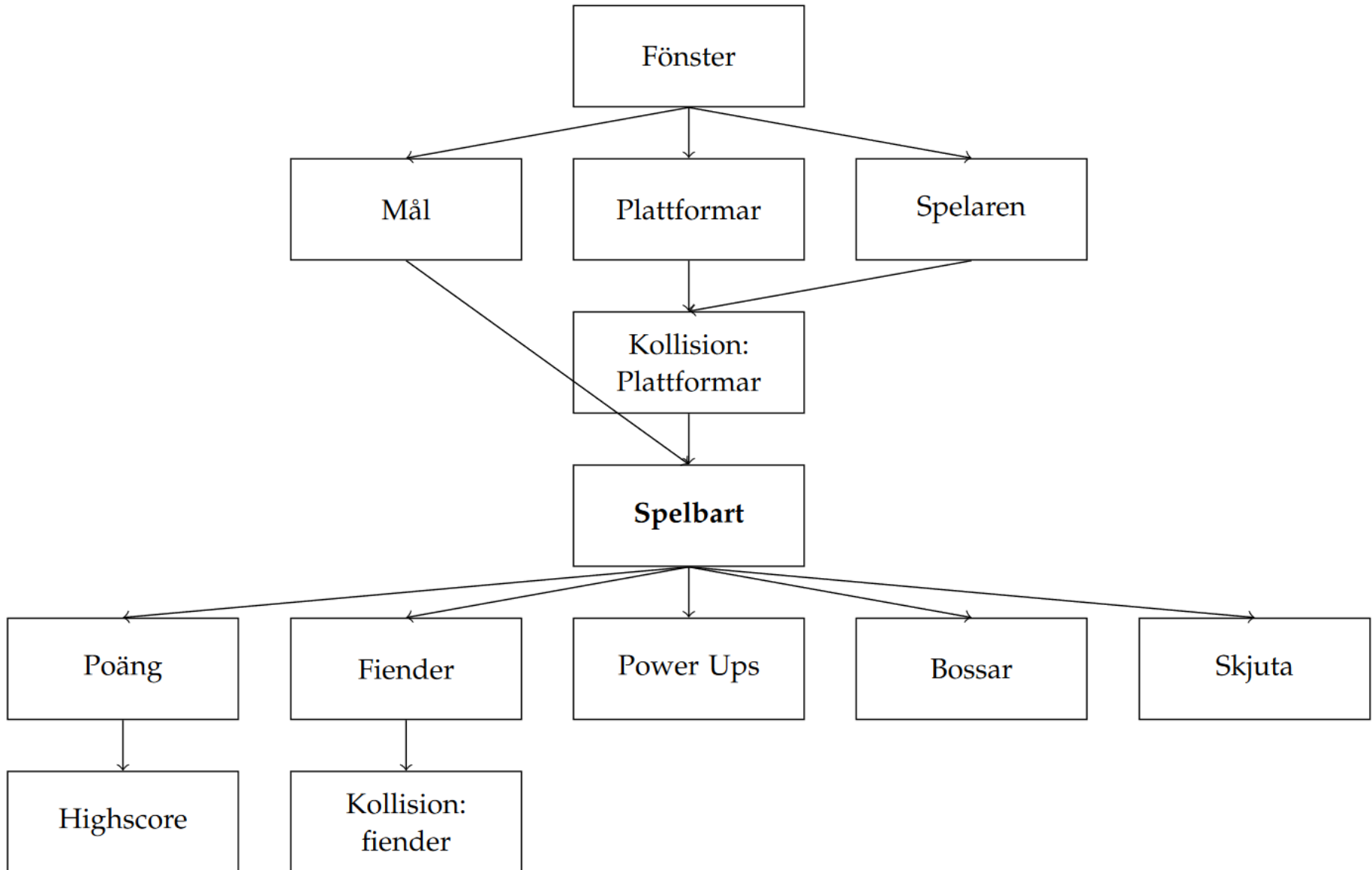
**Hit hinner vi säkert inte!
(Men vem vet...)**

**Inspirationsprojekt har
en generell outline**

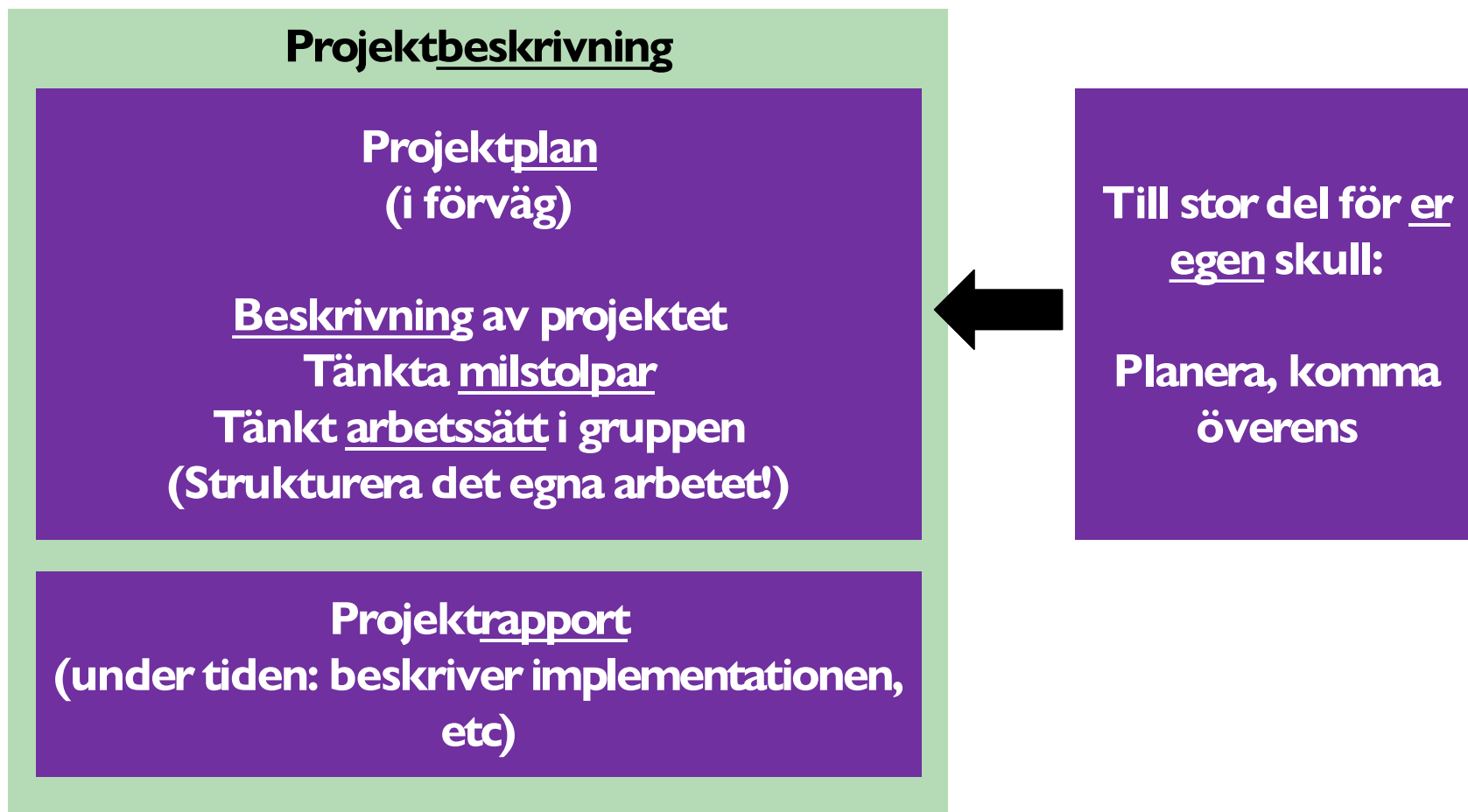
**Behöver utökas,
fler milstolpar,
fler detaljer!**

**Bör justeras efter egna
önskemål**

Lista... eller träd?



- Strukturen hos en projektbeskrivning med en plan...
 - Mall med instruktioner finns på projektets websidor



Projektplan: Studenternas åsikter 1



- [Projektplanen] var annars ett bra hjälpmedel med tanke på planeringen som skrevs innan programmerandet började. Jag hade skrivit bra milstolpar som jag följde till stor del och som var till stor hjälp.
- Vi har använt [projektplanen] ganska mycket för att pricka av mot milstolparna hur långt vi hade kommit med projektet. Detta har varit till stor hjälp för planering av tiden.
- Det är viktigt att börja i god tid och att ha en plan för arbetet klar eftersom att det underlättar arbetet väldigt mycket.
- Vi har lärt oss att man ska lägga ner mycket tid på tänka ut hur man ska bygga upp projektet innan man börjar koda. Om man tänkte igenom allt ner till minsta detalj sparar man mycket tid när man sedan ska koda det.
- The project [plan] was great at the beginning of the project because you had to really think about how you would structure the code. The best thing was milestones since it helped a lot later in the project. You always knew what to do next and it helped to split the project into smaller parts.

Projektplan: Studenternas åsikter 2

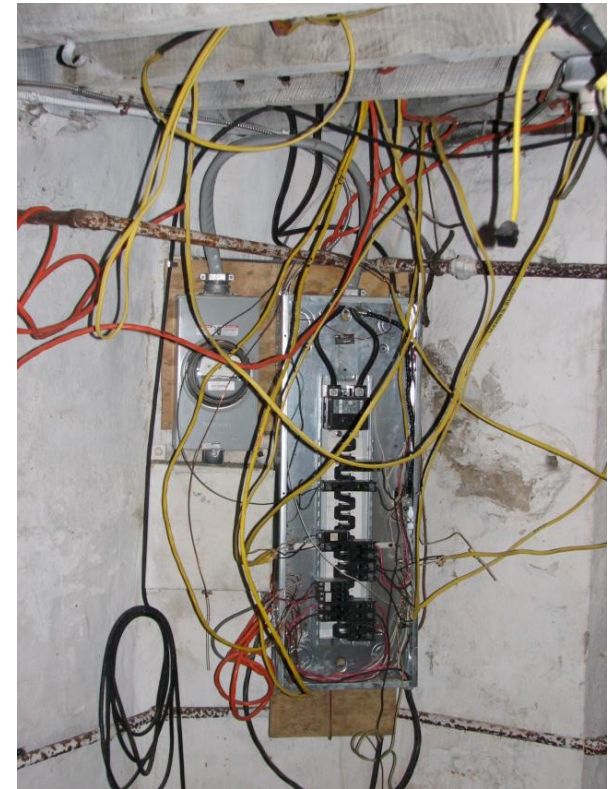


- Vi har inte haft någon nytta av [projektplanen], eftersom man började med det man tyckte behövdes mest. Hade vi vetat i förväg exakt hur vi skulle göra så hade vi följt den men nu hade vi inte det.

Att genomföra ett projekt

- Mål:
 - Visa upp vad ni kan inom objektorientering och Java

- Medel:
 - Skriv ett eget projekt
 - Välskriven kod av hög kvalitet
 - Använd objektorienteringen och Javas finesser
 - Demonstrera vad ni kan inom kursens mål
 - Tänk på vad man ser när man läser koden
 - Inte vad programmet utför, utan *hur*
 - Dokumentera
 - Mer att skriva i projektrapporten – visar hur ni *tänker*, som på en tenta där man inte bara skriver slutliga svaret



- Det tar **längre tid** än man tror
- Räkna med att allt tar **längre tid** än vad man tror
- Start on the project early. It will take **a lot of time** so you can't only work the last days before deadline, especially not if you aim for a high grade.
- Gör projektet. **Vänta inte**, var helst klar till första redovisning. Undvik vänta till omtenta-p.

**Fixa ändå Tetris före projektet,
annars missar ni nödvändig kunskap!**

Det kan också vara en bra idé att fråga assistenterna om hjälp om man kör fast, istället för att sitta och slå huvudet i bordet tills man tappar motivationen att fortsätta.

■ Måla inte in er i ett hörn!

- Ständig uppföljning med handledare
 - Diskutera problem och lösningar
- Be handledaren titta på koden
 - Fånga upp problem tidigt
- Jobba inte enbart hemma!



- Gör projektet steg för steg
 - Lägg till lite mer funktionalitet
 - Se till att det fungerar
 - Dokumentera
 - Putsa och förbättra
 - Kontrollera (inspektioner, ...)
 - Checka in resultatet *ofta*
 - Repetera...

Det var inte förrän slutet som vi gjorde kodinspektioner, som till vår fasa visade
1,661 varningar
(ungefär en tredjedel magiska konstanter).
De flesta av dessa är nu fixade, men vi borde absolut kollat på detta under tiden vi skrev programmet istället.

Tips till nästa års studenter skulle nog vara just att skriva på
[projektrapporten] under tiden som ni programmerar. Det tror jag ökar chansen till ett högre betyg.

Däremot borde jag ha skrivit på resten av
[projektrapporten]
medan jag skrev koden.

- Steg för steg → kan införa *feature freeze* innan slutet
 - Ger tid till **slutlig finputsning** – kvaliteten räknas...
 - Gå genom kodinspektioner
 - Se över *hela* strukturen
 - Uppdatera gammal kod, nu när ni vet bättre
 - Ta bort "dubletter", hitta ineffektiv kod
 - Uppdatera kommentarer och dokumentation
 - Slutför projektrapporten
 - ...

Redovisning, inlämning, deadlines



TDDD78 / U

**Slutet av läsperioden:
Demo 250311, inlämning 250312**

**Fortsätta i tenta-p?
Demo 2503xx, inlämning 250330**

**Jobba vidare? Demo + inlämning i
juni, augusti, oktober, januari**

TDDE30 / D

**Slutet av läsperioden:
Demo 250522, inlämning 250525**

**Jobba vidare?
Tillfällen i augusti, oktober, januari**

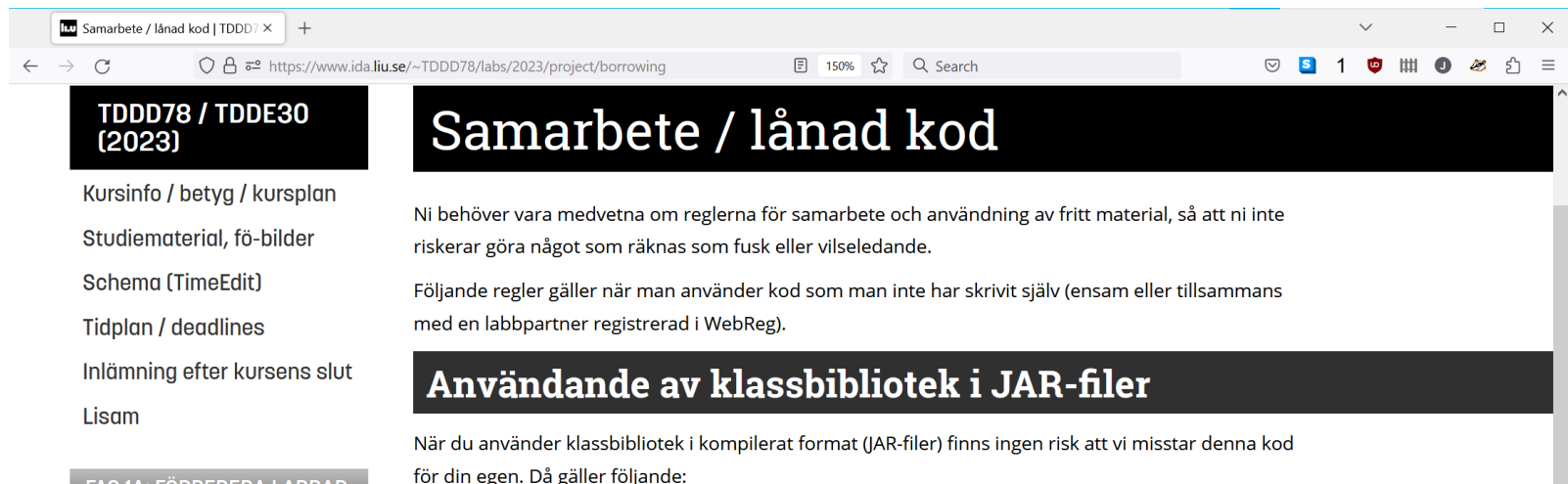
- **Demonstration**
 - Visa vad ni gjort
 - Visa att ni *förstår* vad ni gjort
- **Vid inlämning:**
 - Följ instruktionerna – annars får ni en retur!

Att arbeta med andras kod – undvik fusk!

Korrekt arbete med andras kod

- Läs <https://www.ida.liu.se/~TDDD78/labs/2025/project/borrowing> för att undvika fusk!
 - *Partiell sammanfattning:*
 - Får använda andra klassbibliotek, så länge som er kod *driver implementationen*
 - Får använda okompilerad källkod från andra, om den *markeras korrekt* så *verktyg kan upptäcka den*
 - Får diskutera med andra, om ni *lärt er och förstår* så ni sedan kan *skriva egen kod på egen hand*
 - **Misstanken att det skulle kunna vara fusk måste anmälas!**

Checka in och pusha ofta → en loggad historik över mellanstegen i er kod!



TDDD78 / TDDE30 (2023)

Kursinfo / betyg / kursplan

Studiematerial, fö-bilder

Schema (TimeEdit)

Tidplan / deadlines

Inlämning efter kursens slut

Lisam

Samarbete / lånad kod

Ni behöver vara medvetna om reglerna för samarbete och användning av fritt material, så att ni inte riskerar göra något som räknas som fusk eller vilseledande.

Följande regler gäller när man använder kod som man inte har skrivit själv (ensam eller tillsammans med en labbpartner registrerad i WebReg).

Användande av klassbibliotek i JAR-filer

När du använder klassbibliotek i kompilerat format (JAR-filer) finns ingen risk att vi misstar denna kod för din egen. Då gäller följande:

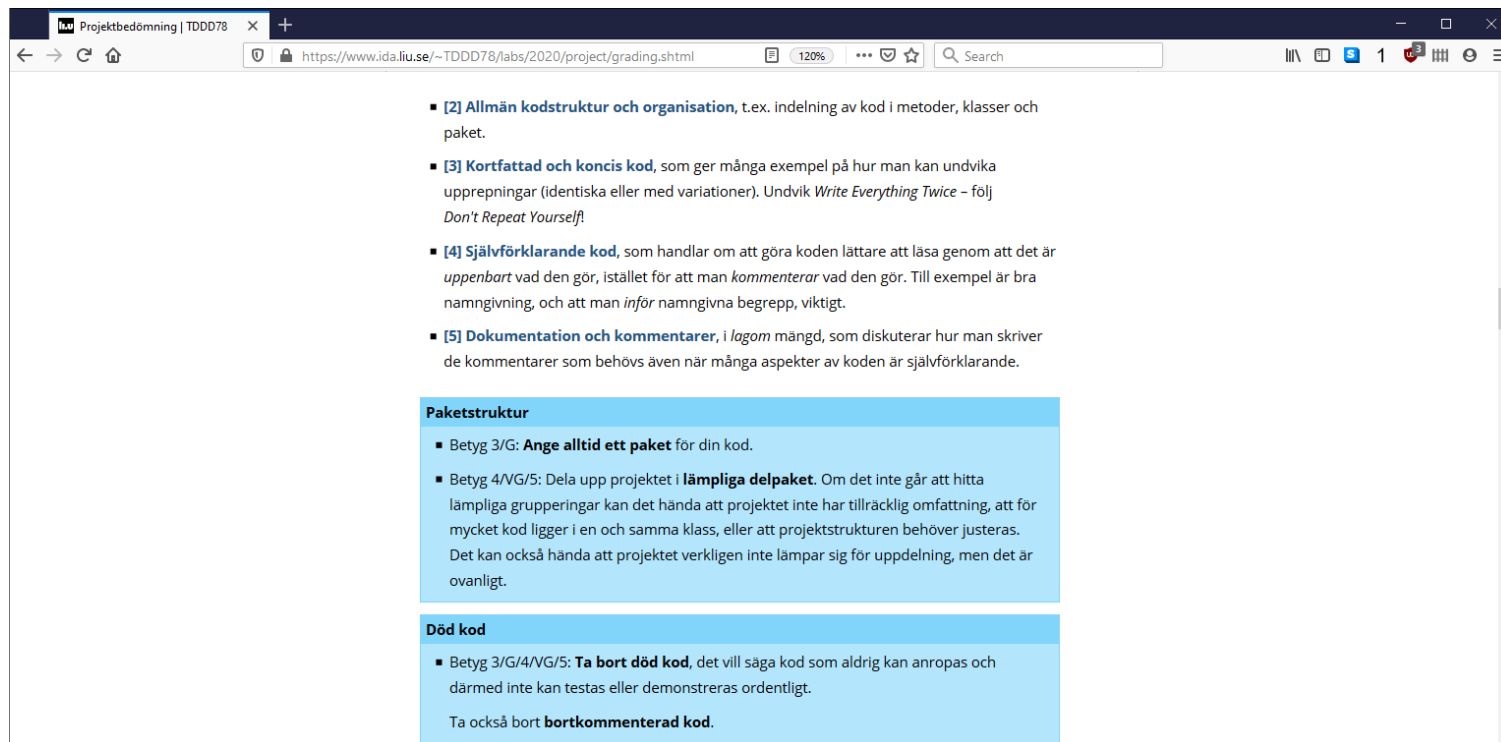
Betygskriterier

- Flera kvalitativa kriterier för betygsättning (3/4/5, G/VG):
 - Objektorientering och Java
 - Använd bra objektorienterade lösningar
 - Visa att ni har förstått OO-principer
 - Följ rekommendationer från föreläsningar:
Hur man skriver *bra* kod, som gör på *rätt* sätt
 - Läsbarhet och struktur
 - Skriv tydlig, lättläst, rimligt dokumenterad kod
 - Omfattning motsvarande 80 timmar/person
 - Sekundärt kriterium – ändå viktigt
 - Ju mindre projekt, desto mer tid för finputsning → högre kvalitet krävs

Kvalitativa → bedömningsfråga
Se websidorna för mer vägledning!

- Även några mer mätbara krav för specifika betyg
 - Paketstruktur, namngivning, felhantering, upprepning, modellering av uppräkningsbara värden, resurshantering, ...

Se websidorna för detaljer!



Projektbedömning | TDDD78

https://www.ida.liu.se/~TDDD78/labs/2020/project/grading.shtml

- [2] **Allmän kodstruktur och organisation**, t.ex. indelning av kod i metoder, klasser och paket.
- [3] **Kortfattad och koncis kod**, som ger många exempel på hur man kan undvika upprepningar (identiska eller med variationer). Undvik *Write Everything Twice* – följ *Don't Repeat Yourself*!
- [4] **Självförklarande kod**, som handlar om att göra koden lättare att läsa genom att det är *uppenbart* vad den gör, istället för att man *kommenterar* vad den gör. Till exempel är bra namngivning, och att man *inför* namngivna begrepp, viktigt.
- [5] **Dokumentation och kommentarer**, i *lagom* mängd, som diskuterar hur man skriver de kommentarer som behövs även när många aspekter av koden är självförklarande.

Paketstruktur

- Betyg 3/G: **Ange alltid ett paket** för din kod.
- Betyg 4/VG/5: Dela upp projektet i **lämpliga delpaket**. Om det inte går att hitta lämpliga grupperingar kan det hända att projektet inte har tillräcklig omfattning, att för mycket kod ligger i en och samma klass, eller att projektstrukturen behöver justeras. Det kan också hända att projektet verkligen inte lämpar sig för uppdelning, men det är ovanligt.

Död kod

- Betyg 3/G/4/VG/5: **Ta bort död kod**, det vill säga kod som aldrig kan anropas och därmed inte kan testas eller demonstreras ordentligt.

Ta också bort **bortkommenterad kod**.

Ignorera inte – vi ger komplettering

Labb 1–4

**Moment
LABx
i LADOK**

**3 hp,
betyg G**

Tetris-4, Tetris-5 – *Behövs för betyg 4/5 på kursen*

**Moment
PRAx
i LADOK**

**Projektet
och projektrapporten**

**3 hp,
betyg 3-
4-5**

TDDE30: Språk och struktur för rapport, 1 hp

1 hp, G

TDDE30: Rapportgranskning

- D (TDDE30): 1 hp för språk- och strukturgranskning
 - (U har detta i en annan kurs)
- Ger **26 timmar extra**
 - **Skriva** mer (projektrapport)
 - **Gå genom** eget språk och struktur i förväg
 - Rekommenderat att **lämna in** 2504xx(?) till granskare på IKOS
 - **SEPARAT INLÄMNING! LISAM!**
 - Tidigt, så man får basera rapporten på det man *har* och *planerar*
 - Ger möjlighet till återkoppling under kursens gång – ofta komplettering!
 - Bedömning återfås 25051x
 - **Projektdemo** 250522
 - **Projektinlämning** 250525 – kod och rapport till ”Java-granskning”
 - Vi använder rapporten för att förstå projektet
 - **Rapportinlämning** 250525 – nästa chans att lämna in till IKOS
 - Komplettering i augusti, oktober men *inte* i januari – nästa omgång maj 2025

Programmering: Vad är vårt fokus?

När är ett program (en inlämning) bra?

När körningen "ger rätt resultat"?

Nej!

När programkoden är tydlig och välstrukturerad,
och visar att ni vet vad ni gör!

Hjälp: Kodanalysen

Ger vägledning, men kan aldrig säga allt

JavaWarnings.java: 0 fält, 2 metod(er)

22List names = new ArrayList();

oo-**RawUseOfParameterizedType** 2 instanser ✖

- A parameterized (generic) type such as `List<ElementType>` is used without providing an actual element type (only `List`). This is supported in Java for backward compatibility but should never be used in new code, as type safety is reduced.
- This is considered a serious issue and must definitely be fixed before handing in a project or lab.
- Raw use of parameterized class `List`

oo-**RawUseOfParameterizedType**

- Raw use of parameterized class `ArrayList`

programming-**MismatchedCollectionQueryUpdate** 1 instans ✖

- A collection is only updated without being queried, or vice versa. Can indicate a bug (why would we add objects without using them?), or incomplete code, or preparation for future features. In either case, it is typically unknown whether the correct objects are added.
- Contents of collection `names` are updated, but never queried

23names.add(name);

oo-**UNCHECKED_WARNING** 1 instans ✖

- This warning can occur in several situations.
- Unchecked assignment* can for example mean that you are assigning a value of a 'raw' generic type, such as `List`, to a variable that specifies an element type, such as `List<String>`. Make sure that you specify element types everywhere (or in some cases the 'diamond' `<>`, meaning 'the same element types that the variable already specifies').
- Another situation is where a dynamic cast executed at runtime cannot actually check whether the object has the

- Ser det ut så här,
blir vi misstänksamma...

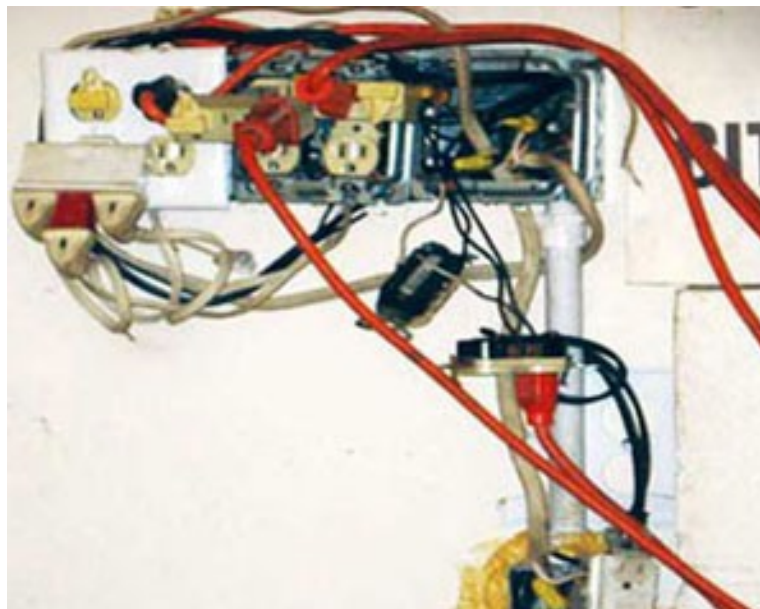
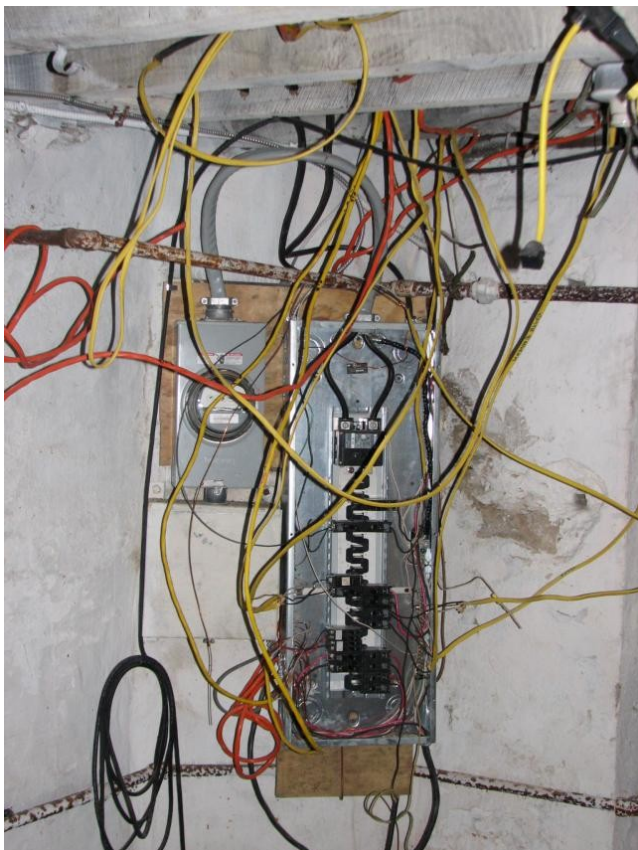


- Har ni kopplat så här, är det farligt...



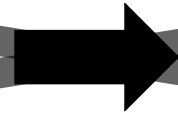
Kvalitet (3)

- Ser det ut så här, undrar vi vad ni egentligen har gjort
 - **Även** om ni faktiskt fick lampan att lysa



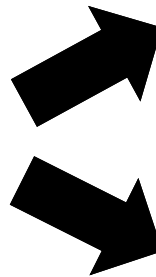
Orsak och verkan

Programmet fungerar
(ger rätt utmatning, ...)



Jag kan programmera!

Jag kan programmera!



Programmet fungerar
(ger rätt utmatning, ...)

Programmet är välskrivet,
lätt att förstå,
välstrukturerat, ...

Jag förstår varför
jag gör på ett visst sätt

- Skriv **bra** kod!
 - Lättläst
 - Bra namngivning
 - Strukturerad, modulär
 - Vældokumenterad och kommenterad – där det behövs mest!
- Extremt viktigt för:
 - Hur programmet kan utökas i framtiden
 - Hur andra kan förstå programmet
 - Hur robust det är när fel uppstår
 - ...

Utveckling är ofta 20% av arbetet...

Underhåll är 80%!

Hur ska man skriva kod?

- **Tips:**

- **“Always code as if the person who ends up maintaining your code is a violent psychopath who knows where you live.”**



- *"I usually maintain my own code, so the as-if is true."*
– <http://c2.com/cgi/wiki?CodeForTheMaintainer>