|  |
| --- |
| Kungliga Tekniska Högskolan |
| Simulering av smittspridning rapport |
|  |
|  |
| **Sadok Habibi Dalin, 19940519-6974. Alexander Barosen, 19940205-5710** |
| **2015-11-25**  [**Sadokd@kth.se**](mailto:Sadokd@kth.se)**, Abarosen@kth.se**  **Ingenjörskunskap och ingenjörsrollen ICT II1304** |

Innehållsförteckning

[1. Introduktion 3](#_Toc436228855)

[2. Smittmodellen 3](#_Toc436228856)

[2.1 Smittmodellens egenskaper 3](#_Toc436228857)

[2.2 Reflektion kring vad som är bra, mindre bra med modellen 3](#_Toc436228858)

[3. Beskrivning och validering av implementationen 4](#_Toc436228859)

[3.1 Beskrivning av implementationen i text 4](#_Toc436228860)

[3.1.1 Klassen Human 4](#_Toc436228861)

[3.1.2 Klassen Contamination 4](#_Toc436228862)

[3.1.3 Klassen DoHundred 6](#_Toc436228863)

[3.2 Beskrivning av implementationen med hjälp utav modellverktyg 6](#_Toc436228864)

[3.3 Validering av implementationen 7](#_Toc436228865)

[4. Validering/Verifiering av arbetsprocessen 10](#_Toc436228866)

[5. Val av experiment 10](#_Toc436228867)

[6. Resultat 11](#_Toc436228868)

[7. Slutsats 13](#_Toc436228869)

[8. Kör-instruktioner och sökväg till källkoden på KTH servrarna 13](#_Toc436228870)

[Appendix A: Kravanalys 14](#_Toc436228871)

[Appendix B: Disposition/Mall 19](#_Toc436228872)

[Appendix C: Väldokumenterad källkod 21](#_Toc436228873)

[Appendix D: Slumptal som användes vid experimenten 31](#_Toc436228874)

[Appendix E: Rådata från körningar 35](#_Toc436228875)

[Appendix F: Individuella reflektioner över arbetsgången 39](#_Toc436228876)

[F.1 Alexander Barosens reflektioner över arbetsgången 39](#_Toc436228877)

[F.2 Sadok Habibi Dalins reflektioner över arbetsgången 39](#_Toc436228878)

# 1. Introduktion

Sjukdomar och smittor är vanliga åkommor i världen. För att förutsäga sjukdomars spridning ställs olika modeller upp. Eftersom modellerna för att modellera sjukdomsspridning ofta är komplexa, utmynnar de i simuleringar i med stokastiska funktioner. I vårt fall simulerar vi en enkel smittmodell.

Vid utveckling av metoder för att förhindra smittspridning är sjukdomars smittspridning av intresse. En sjukdom som lätt utvecklas till en epidemi är ett betydligt större hot och kräver andra åtgärder än en sjukdom med låg smittsannolikhet. Ett av problemen är dock att hitta smittsannolikheten då en sjukdom utvecklas till en epidemi.

I den här rapporten analyserar vi utdata från ett antal simuleringar av en enkel smittmodell. Vårt mål med rapporten är att hitta vid vilken smittsannolikhet då sjukdomen utvecklas till en epidemi.

I Kapitel 2 beskrivs smittmodellen och dess egenskaper diskuteras. I Kapitel 3 beskriver vi implementationen och hur den har validerats. I kapitel 4 beskrivs arbetsprocessen. I Kapitel 5 redovisar vi de experiment vi har utfört. I Kapitel 6 presenterar vi våra resultat. I Kapitel 7 analyserar vi de sammanställda resultaten och drar slutsatser. Slutligen i Kapitel 8 beskriver vi hur programmet kan köras och var källkoden ligger på KTH’s shell-servrar.

# 2. Smittmodellen

## 2.1 Smittmodellens egenskaper

Populationen i smittspridningsmodellen ska modelleras som en N\*N matris där varje element i matrisen innehåller en individ. Sjukdomen sprids via direktkontakt mellan individer det vill säga bara till de 8 direkta grannarna från en viss individ i matrisen. En individ är smittad i en dag och sedan sjuk i ett slumpmässigt antal dagar uniformt fördelar i ett intervall. Varje dag finns det en viss sannolikhet att en varje sjuk individ dör. En död individ kan inte smittas eller smitta andra. En sjuk individ kan inte smittas. En smittad individ kan varken dö, smittas eller smitta andra. Ingen individ kan bli smittad mer än 1 gång under en simulering, det vill säga en frisk individ som har varit sjuk kan inte bli smittad igen. En körning av simuleringen ska pågå till dess att inga sjuka återstår i populationen.

## 2.2 Reflektion kring vad som är bra, mindre bra med modellen

En styrka hos modellen är att den är relativt flexibel. Sjukdomar som smittar via direktkontakt och som har olika smittsannolikhet, dödssannolikhet och tillfriskningstid kan modelleras vilket innebär att man kan modellera många vanliga sjukdomar. Modellen är också enkel implementera och verifiera p.g.a. dess simplicitet.

Modellen har flera begränsningar och svagheter. Distansen mellan människor är inte dynamisk utan de antas alltid vara näst intill varandra. Inga individer i populationen kan förflytta sig utan bibehåller samma position under en simulering. Sjukdomen kan inte smitta via andra källor än direkt kontakt. I verkligheten kan sjukdomar sprida sig via djur, insekter, vatten och luft. Olika sjukdomar sprider sig också olika lätt i olika miljöer, vilket inte tas i beaktning i modellen. Mänsklig resistans modelleras inte. Sjukdomar som är relativt farliga och smittsamma brukar det utvecklas vaccin eller medicin för. Vid stora epidemier brukar också strängare sanitetsregler införas vilket minskar sannolikheten för smittspridning. I modellen är smittsannolikheten statisk under en simulering. I verkligheten kan döda människor också sprida en sjukdom(se Ebolafeber).

Modellen kan enkelt utökas så att det kan distansen kan variera mellan människor genom att t.ex. modellera vatten eller obefolkad mark. Vatten och obefolkad mark kan modelleras som tomma element i matrisen. Man kan utveckla modellen så att sjukdomen sprider sig en slumpmässig distans i olika riktningar för att modellera luftburna sjukdomar. Olika miljöer kan modelleras i själva matrisen genom att olika element i matrisen har en modifikationskoefficient för smittsannolikheten. Vaccin och medicin utveckling kan modelleras genom att alla immuna människor har en viss sannolikhet att per dag utveckla ett vaccin. När de har utvecklat ett vaccin sprider de det till sina närmaste grannar per dag som i sin tur sprider till sina grannar.

# 3. Beskrivning och validering av implementationen

## 3.1 Beskrivning av implementationen i text

Klasserna Human och Contamination utgör implementationen av smittmodellen och uppfyller alla krav satta på implementationen. Klassen DoHundred är en utökning av implementationen och skrevs för att enklare kunna köra flera simuleringar och generera andra utdata. Nedan följer en detaljerad beskrivning av klasserna Human, Contamination och DoHundred.

## 3.1.1 Klassen Human

Ett objekt av klassen Human representerar en enskild individ. Det kändes naturligt att modellera en individ som ett objekt eftersom en individ i modellen har flera dynamiska attributer som hälsostatus och tillfriskningstid.

Hälsostatusen för en individ modelleras som en enum HealthStatus det vill säga som en variabel som kan sättas lika med en konstant ur en fördefinierad mängd med konstanter. Detta för att göra det enklare att förstå och modifiera koden. Det är i min åsikt enklare att relatera ett ord till en viss hälsostatus än en siffra till en viss hälsostatus. Tillfriskningstiden mäts i hela dagar och modelleras således som ett heltal.

Metoderna getStatus, setStatus, getSickTime och decrementSickTime är i min åsikt självförklarande. Metoden setSick sätter en individs status till ”sick” och dess tillfriskningstid lika med argumentet sickTime. Kill metoden sätter en individs hälsostatus till död. Inga av instansvariablerna behöver återställas till skönsvärden eftersom ingen individ kan bli smittad mer än 1 gång.

## 3.1.2 Klassen Contamination

En körning av Contamination är ekvivalent med en simulering. De globala variablerna dailyRecovered, dailyDeaths och dailyInfected används för att generera utdata för varje simulerad dag. Den globala variabeln infected används för att kontrollera när simuleringen ska ta slut. Den globala pseudo slumptalsgeneratorn rand används för att bestämma när en individ ska bli smittad, dö och gränserna för dess tilfriskningstidsintervall. Alla nämnda variabler är globala för att de används i flera metoder och för att koden blir lättare att läsa.

Metoden main utför en körning av simuleringen. I början av metoden sätts skönsvärden för smittsannolikheten, populationens storlek, minimala antalet dygn som en individ kan vara sjuk, Maximala antalet dygn som en individ kan vara sjuk och dödssannolikheten. Därefter följer en kombination av if-satser som tar in olika många kommandoradargument. Kommandoradargumenten kan användas för att modifiera alla parametrar utom sjuka individers antal och placering i populationen. Om inga kommandoradargument finns får användaren via konsolen välja om den vill köra en simulering med skönsvärden eller om den vill modifiera dem.

Tidens gång modelleras med en while-slinga. En upprepning av slingan representerar en dag i modellen. För varje iteration uppdateras populationens status via ett anrop till metoden update. Efter det uppdateras utskriftsvariablerna och skriva ut. Till sist återställs alla dagliga utskriftsvariabler. När while-slingan är slut är simuleringen över.

Metoden primaryTest är snarlik main förutom att inga utskrifter sker i den. Metoden har också ett annat returvärde, ackumulerat antal infekterade under en körning. primaryTest anropas endast från main metoden i klassen DoHundred och finns för att generera kompakta råa resultat vid ett stort antal simuleringar.

setRandomSeed metoden ändrar slumptalsfröet för pseudo slumptalsgeneratorn rand som används för att bestämma när en individ ska bli smittad, dö och gränserna för dess tilfriskningstidsintervall.

Metoden initialize skapar en tvådimensionell array givet en höjd/bredd N och fyller alla element med friska individer. Metoden skrevs för att förtydliga koden och för att undvika duplicerad kod i main och primaryTest.

Metoden update uppdaterar populationens status för en dag. 2 nästlade for-slingor används för att iterera genom varje individ i populationsmatrisen rad för rad i stigande ordning. För varje enskild individ utförs flera operationer. Om individen är sjuk genereras ett slumptal mellan 0 och 1 som jämförs med dödssannolikheten. Om det genererade slumptalet är mindre än dödssannolikheten sätts individens status till död. Om individen inte dör anropas metoden infect för alla 8 direkta grannar till individen där man undersöker om grannen ska bli smittad eller inte. Efter det dekrementeras individens tillfriskningstid. Om individens tillfriskningstid då är mindre eller lika med 0 sätts individens status till immun.

Efter att vi har itererat genom alla individer i populationen och bestämt vilka som ska bli smittade itererar vi genom alla individer på samma sätt igen för att göra alla smittade till sjuka. För varje enskild individ kollar vi då om individen är smittad. Om den är det sätts individens status till sjuk och dess tillfriskningstid sätts med hjälp av pseudo slumptalsgeneratorn till ett heltal mellan tillfriskningstidsintervallets gränser. Update skrevs i en separat metod för att undvika duplicerad kod i primaryTest och main.

Metoden infect undersöker om en viss frisk människa som är innanför populationsmatrisens gränser ska bli infekterad. Den gör detta genom att undersöka om ett slumpmässigt decimaltal uniformt fördelat i intervallet [0,1] är mindre än smittsannolikheten. Om den är det sätts individens status till smittad. Metoden finns för att slippa duplicerad kod när vi itererar genom en individs direkta grannar.

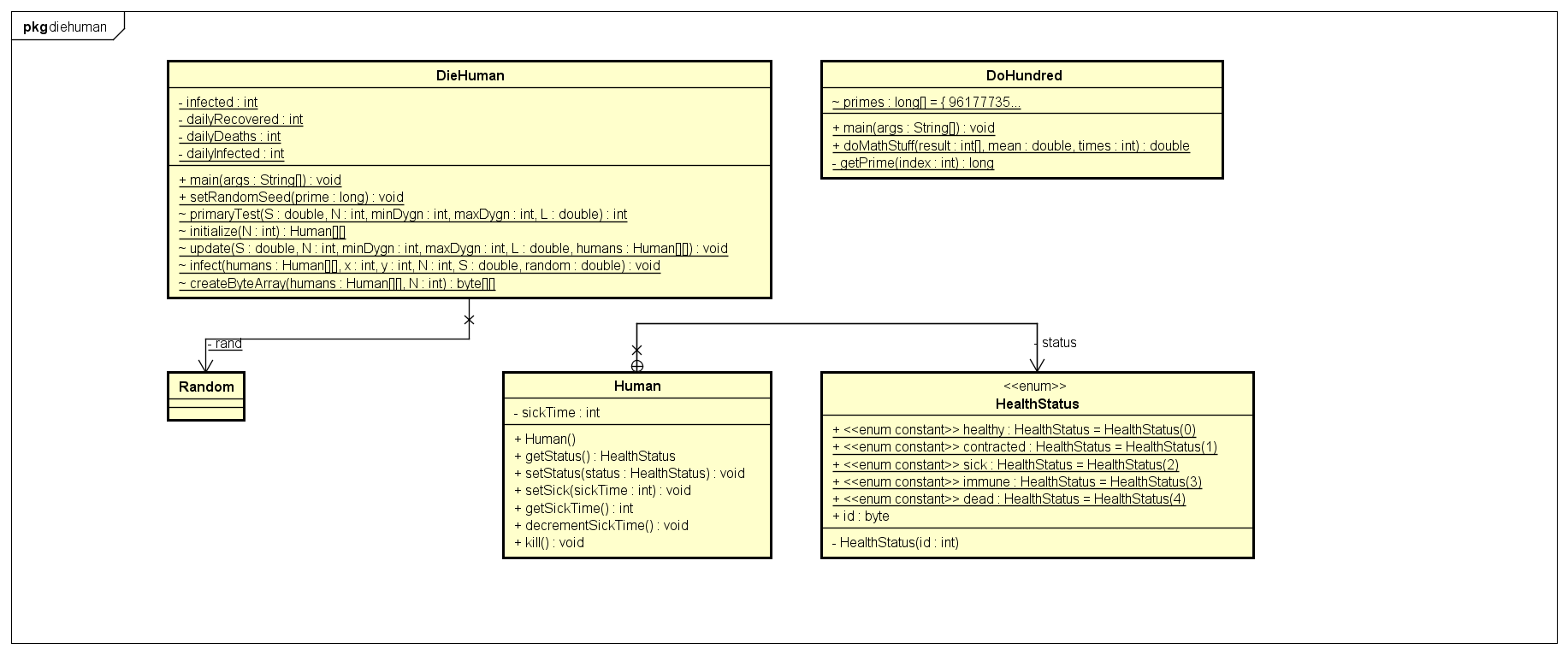
## 3.1.3 Klassen DoHundred

En körning av DoHundred kör upp till 100 st simuleringar för en viss smittsannolikhet, dödssannolikhet, gränser på tillfriskningstidsintervallet och populationsstorlek. En variabel times i main metoden används för att bestämma antal simuleringar som ska köras. Resultatet är medelvärdet av ackumulerat antal infekterade efter times antal körningar och standardavvikelsen av ackumulerat antal infekterade. Alla parametrar förutom antalet infekterade och deras position i populationsmatrisen kan tas in via kommandoraden.

Simuleringarna görs genom att för varje upprepning i en while-slinga lagra returvärdet från primaryTest i klassen Contamination i en array results av storlek times. While-slingan upprepas times gånger. För varje iteration i slingan sätts ett slumptalsfrö för slumptalsgeneratorn rand i klassen Contamination till ett unikt primtal. Primtalen är lagrade i en global statisk array av storlek 100 och kan nås via funktionen getPrime. I while-slingan ackumuleras även totala antalet smittade under alla simuleringar i en variabel accInfected. accInfected delas med times för att få det aritmetiska medelvärdet för antalet smittade per simulering. Det aritmetiska medelvärdet skrivs till sist ut tillsammans med stickprovets standardavvikelse.

Metoden sampleStandardvariation tar resultatarrayen results samt det aritmetiska medelvärdet för antalet smittade per simulering mean och räknar ut stickprovets standardavvikelse .

## 3.2 Beskrivning av implementationen med hjälp utav modellverktyg

UML har använts för att beskriva implementationen. I figur 1 illustreras ett klassdiagram över alla klasser i implementationen. 

*figur 1: Klassdiagram för varje klass i implementationen.*

## 3.3 Validering av implementationen

Implementationen har validerats huvudsakligen med ”glass box” testning. Generella enhetstester och integrationstester har skrivits för flera metoder i klasserna Human, Contamination och DoHundred. Systemtester för specifika extrema och normala värden på in-parametrar har dokumenterats och genomförts. Nedan följer en beskrivning av testfallen där inga specifika in-parametrar anges.

**Slumptalsgenerering:**

* **Moduler:** funktionen nextInt i klassen Random.
* **Indata:** -
* **Förväntat resultat:** Ett heltal uniformt fördelat mellan och .
* **Moduler:** funktionen nextDouble i klassen Random.
* **Indata:** -
* **Förväntat resultat:** Ett decimaltal uniformt fördelat mellan 0 och 1.

**Klassen Human:**

* **Moduler:** Konstruktor.
* **Indata:** -
* **Förväntat resultat:** En frisk individ det vill säga ett objekt av human med status = healthy och sicktime = 0.
* **Moduler:** Metoden setStatus.
* **Indata:** -
* **Förväntat resultat:** En frisk individ det vill säga ett objekt av human med status = healthy.
* **Moduler:** Metoden getStatus.
* **Indata:** -
* **Förväntat resultat:** En individs status i form av en enum. Metoden borde returnera “healthy”, “immune”, “sick”, “contracted” eller “dead”.
* **Moduler:** Metoden setSick.
* **Indata:** Ett positivt heltal sickTime.
* **Förväntat resultat:** Individens status bör ändras från ”healthy” till ”sick”. Individens tillfriskningstid bör sättas till sickTime.
* **Moduler:** Metoden decrementSickTime.
* **Indata:** -
* **Förväntat resultat:** Individens tillfriskningstid bör dekrementeras d.v.s. instansvariabeln sickTime bör minska med ett.

**Klassen Contamination:**

* **Moduler:** Metoden setRandomSeed.
* **Indata:** Ett 64 bitars heltal prime.
* **Förväntat resultat:** Individens tillfriskningstid bör dekrementeras d.v.s. instansvariabeln sickTime bör minska med ett.
* **Moduler:** Metoden initialize.
* **Indata:** Ett 32 bitars heltal N större än 0.
* **Förväntat resultat:** Metoden bör returnera en tvådimensionell kvadratisk array med N\*N antal element där varje element innehåller en frisk individ. Med andra ord så ska varje element i arrayen innehålla ett objekt av klassen Human med instansvariabeln status = healthy.
* **Moduler:** Metoden infect.
* **Indata:** En tvådimensionell kvadratisk array humans där varje element innehåller ett objekt av klassen Human; Två heltalsindex x och y för en individ i humans; längden/bredden på humans N; smittsannolikheten S; ett uniformt fördelat decimaltal mellan 0 och 1 random.
* **Förväntat resultat:** Om koordinaterna x, y specificerar en individ som är frisk och innanför matrisens gränser samt att slumptalet random är mindre än smittsanolikheten S borde individens status sättas till ”contracted” d.v.s. individen bör bli smittad.

De testfall där specifika in-parametrar anges skrevs främst för att täcka metoder som har många utfall. Nedan följer en beskrivning av testfallen där specifika in-parametrar anges.

**Metoden update i klassen Contamination:**

* **Moduler:** Metoderna update och infect i klassen Contamination. Metoderna getSickTime, setStatus, setSick, kill, decrementSickTime och getStatus i klassen Human. Metoderna nextInt och nextDouble i klassen Random.
* **Indata:** S = 1, N = 3, minDygn = 3, maxDygn = 3, L = 0, humans innehåller en smittad individ i mitten och friska individer för övrigt.
* **Förväntat resultat:** Efter en körning av metoden bör humans innehålla en sjuk individ i mitten och friska individer för övrigt.
* **Indata:** S = 1, N = 3, minDygn = 3, maxDygn = 3, L = 1, humans innehåller en sjuk individ i mitten och friska individer för övrigt.
* **Förväntat resultat:** Efter en körning av metoden bör humans innehålla en död individ i mitten och friska individer för övrigt.
* **Indata:** S = 0.5, N = 3, minDygn = 3, maxDygn = 3, L = 0, humans innehåller en sjuk individ i mitten och friska individer för övrigt.
* **Förväntat resultat:** Efter en körning av metoden bör ungefär hälften av alla matriser i individen vara smittade.

**Metoden main i klassen Contamination:**

* **Moduler:** Metoderna initialize, update och infect i klassen Contamination. Metoderna getSickTime, setStatus, setSick, kill, decrementSickTime och getStatus i klassen Human. Metoderna nextInt och nextDouble i klassen Random. Metoden parseInt i klassen Integer. Metoden parseDouble i klassen Double.
* **Indata:** S = 1, N = 3, minDygn = 3, maxDygn = 3, L = 0, humans innehåller endast friska människor.
* **Förväntat resultat:** Simuleringen bör inte starta det vill säga endast friska individer bör finnas i matrisen.
* **Indata:** S = 1, N = 3, minDygn = 3, maxDygn = 3, L = 1, humans innehåller en sjuk individ i mitten och friska individer för övrigt.
* **Förväntat resultat:** Efter en körning av metoden bör humans innehålla en död individ i mitten och friska individer för övrigt.
* **Indata:** S = 0, N = 3, minDygn = 3, maxDygn = 3, L = 0, humans innehåller en smittad individ i mitten och friska individer för övrigt.
* **Förväntat resultat:** Efter en körning av metoden bör humans innehålla en immun individ i mitten och friska individer för övrigt.
* **Indata:** S = 1, N = 3, minDygn = 3, maxDygn = 3, L = 0, humans innehåller en sjuk individ i mitten och friska individer för övrigt.
* **Förväntat resultat:** Efter en körning av metoden bör humans innehålla endast immuna individer.
* **Indata:** S = 0.5, N = 50, minDygn = 3, maxDygn = 7, L = 0.1, humans innehåller en sjuk individ i mitten och friska individer för övrigt.
* **Förväntat resultat:** Efter en körning av metoden bör humans innehålla ungefär döda individer. Större delen av populationen bör vara immun.

**Metoden main i klassen DoHundred:**

* **Moduler:** Metoderna doMathStuff och getPrime i klassen DoHundred. Metoderna pow och sqrt i klassen Math. Metoderna initialize, update, primaryTest, setRandomSeed och infect i klassen Contamination. Metoderna getSickTime, setStatus, setSick, kill, decrementSickTime och getStatus i klassen Human. Metoderna nextInt och nextDouble i klassen Random. Metoden parseInt i klassen Integer. Metoden parseDouble i klassen Double.
* **Indata:** times = 1, S = 0, N = 50, minDygn = 3, maxDygn = 3, L = 0, humans innehåller en sjuk individ i mitten och friska individer för övrigt.
* **Förväntat resultat:** Efter en körning av metoden bör det aritmetiska medelvärdet av antalet smittade i alla körningar vara lika med 1. Stickprovets standardavvikelse bör vara lika med 0.
* **Indata:** times = 10, S = 0, N = 50, minDygn = 3, maxDygn = 3, L = 0, humans innehåller en sjuk individ i mitten och friska individer för övrigt.
* **Förväntat resultat:** Efter en körning av metoden bör det aritmetiska medelvärdet av antalet smittade i alla körningar vara lika med 1. Stickprovets standardavvikelse bör vara lika med 0.
* **Indata:** times = 100, S = 1, N = 50, minDygn = 3, maxDygn = 3, L = 0, humans innehåller en sjuk individ i mitten och friska individer för övrigt.
* **Förväntat resultat:** Efter en körning av metoden bör det aritmetiska medelvärdet av antalet smittade i alla körningar vara lika med 2500. Stickprovets standardavvikelse bör vara lika med 0.
* **Indata:** times = 1, S = 0.5, N = 3, minDygn = 3, maxDygn = 3, L = 0, humans innehåller en sjuk individ i mitten och friska individer för övrigt.
* **Förväntat resultat:** Efter en körning av metoden måste det aritmetiska medelvärdet av antalet smittade i alla körningar ligga i intervallet [1, 9]. Stickprovets standardavvikelse måste ligga i intervallet [0, ].

# 4. Validering/Verifiering av arbetsprocessen

Arbetsprocessen bestod utav 2 steg, uppgiftsuppställningen och implementationen. Vi gjorde uppgiftsuppställningen genom att läsa genom labb PM:et och skriva en tolkning av labbuppgiften och smittmodellen. Uppgiftsuppställningen validerades med en kravanalys där vi utförligt beskrev varje krav på implementationen och arbetsmetodiken.

Implementationen gjordes utifrån kravanalysen. Implementationen validerades på flera sätt. Vi skrev bland annat en utförlig beskrivning av implementationen i text och en beskrivning av implementationen med UML. Dessa beskrivningar jämfördes sedan med kraven på implementationen beskrivna i kravanalysen för att hitta diskrepanser. Implementationen validerades också med hjälp utav en heltäckande mängd med enhetstester, integrationstester och systemtester som dokumenterades och utfördes som ”glass box” test.

# 5. Val av experiment

Nedan följer en beskrivning av de tre experiment vi utförde för att få fram tröskelvärdet för smittsannolikheten då sjukdomen utvecklas till en epidemi. Alla experiment är baserade på en population av 50\*50. Minsta antalet dygn för att en person ska bli frisk är tre dagar. Maximala antalet dagar för att en person ska bli frisk är nio dagar. Sjukdomens dödlighet är 0% d.v.s. ingen kan dö av sjukdomen. Alla experiment är baserade på att det initialt finns en smittad person i centrum av befolkningen.

**Experiment Stickprov:**

Detta experiment utfördes för att få en uppskattning av genomsnittliga smittade. Smitt-sannolikheterna 25%, 50% och 75% testades. Detta är för att senare kunna köra experiment med ett mindre och mer exakt intervall. Beroende på hur nära medelvärdena hamnar kring ett tröskelvärde för en epidemi kan man med senare experiment uppskatta ett mer exakt värde. Detta experiment utfördes med klassen Contamination och endast för att få en överblick.

**Experiment Överblick:**

Detta försök är för att hitta smitt-sannolikheten som orsakar 50% infektion i befolkningen, dvs. när sjukdomen orsakar en epidemi. Tidigare experiment visade att en smittsannolikhet på 25% orsakade infektioner på nära hela befolkningens mängd. Därför handlar detta om experiment om att se över intervallet 0% till 25%. För varje procent i heltal utförs hundra körningar. Genom att köra programmet och samla medelvärdena för körningarna kan sedan en graf plottas. Utdata ifrån körningarna är medelvärde och standardavvikelse. Denna data kommer ifrån klassen DoHundred som kör önskat smitt-sannolikhetsvärde hundra gånger. Insamlingen och bearbetningen samt skapandet av graf gjordes i Excel.

**Experiment Tröskelvärde:**

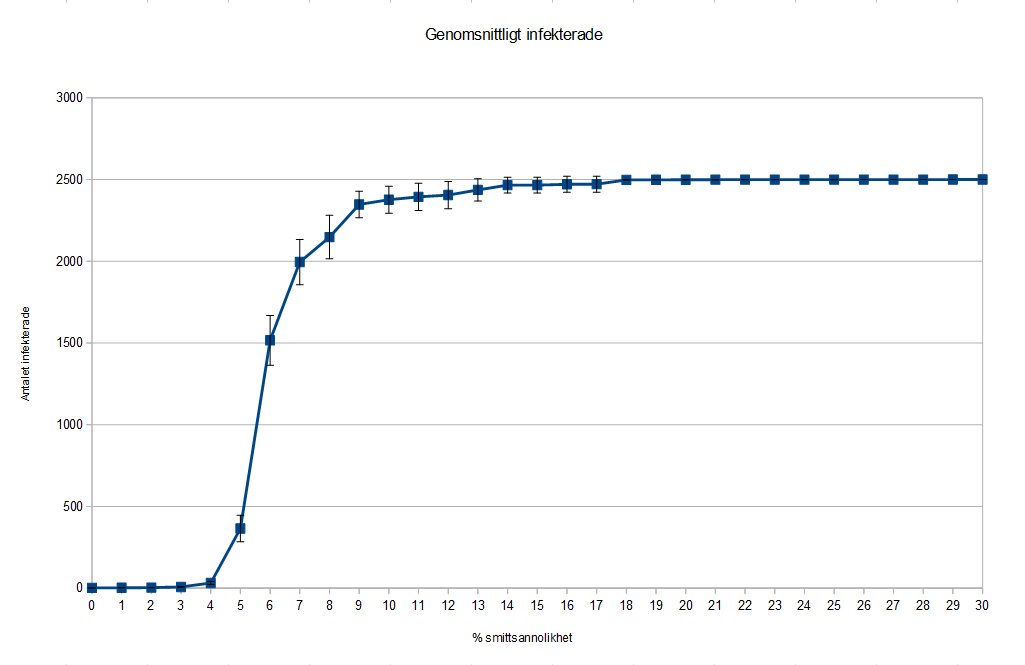
Genom tidigare experiment, dras slutsatsen ifrån dess graf att ett intervall mellan 4% och 7% lär vara intressant att se över mer noggrant. Genom att samla in medelvärdena och standardavvikelsen i ett tätare intervall kan en mer detaljerad graf plottas. Detta är för att ska kunna dra en rimlig slutsats. Konfidensintervall beräknas även utifrån standardavvikelsen givet ifrån körningarna. Insamlingen av data sker för varje 0.05% i intervallet 4% och 7%, då vi enligt tidigare experiment kommit fram till att detta område kan innehålla det sökta tröskelvärdet. Denna data kommer ifrån klassen DoHundred som kör önskat smitt-sannolikhetsvärde hundra gånger. Insamlingen och bearbetningen samt skapandet av graf gjordes i Excel.

Vid beräkning av konfidensintervallen så är standardavvikelsen okänd. Därför skattar vi den med stickprov av mätdata. Eftersom detta stickprov påverkar osäkerheten för beräkningen av konfidensintervallen blir beräkningen av konfidensintervallet över en normalfördelning inte lika korrekt. Därför väljer vi att använda en t-fördelning vid beräkningen då en t-fördelning tar denna osäkerhet i beaktning och konfidensintervallen blir mer korrekta.

# 6. Resultat

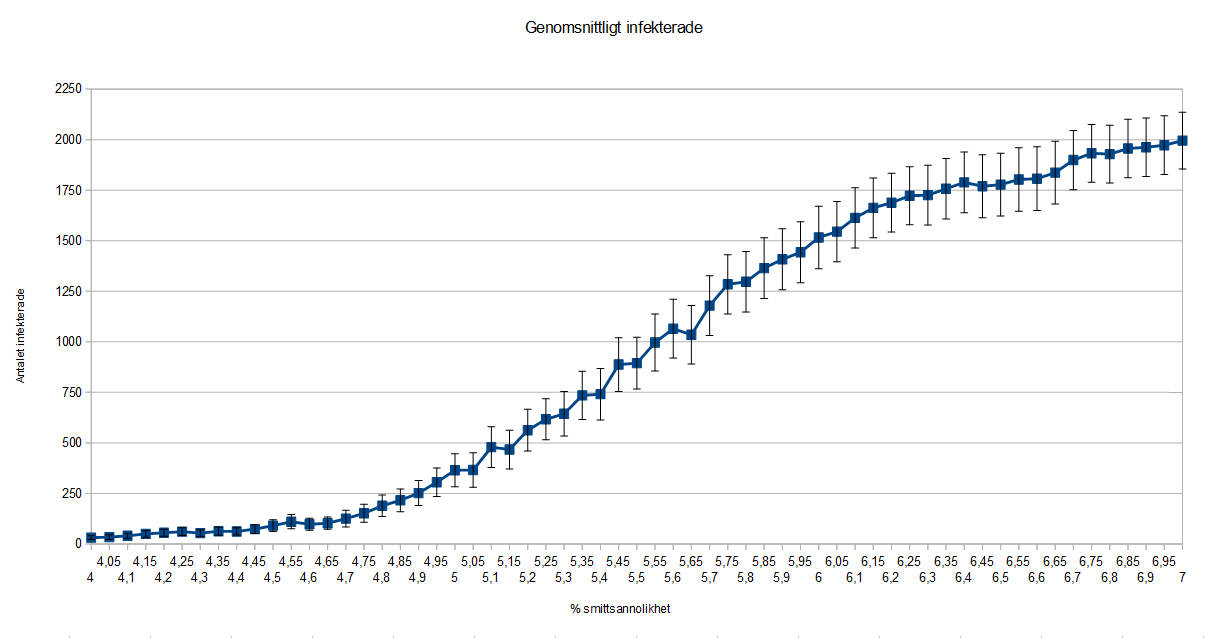
Här presenteras resultaten av simuleringarna. Resultaten presenteras i graferna med antalet infekterade(y-axeln) och smitt-sannolikheten i procent(x-axeln).

Grafen i figur 2 visar en grov uppskattning av var de förväntade värdena borde ligga. Konfidensintervallen är beräknade med konfidensgraden på 95%. Grafen i figur 2 är baserad på resultat från överblicksexperimentet.



*figur 2: Graf som visar konfidensintervallen för antalet smittade under 100 körningar för olika smittsannolikheter i intervallet [0,30].*

Grafen figur 3 visar de smitt-sannolikheter som befinner sig kring 1250 infekterade. Konfidensintervallen är beräknade med konfidensgraden på 95%. Grafen i figur 3 är baserad på resultaten från experiment Tröskelvärde.



*figur 3: Graf som visar konfidensintervallen för antalet smittade under 100 körningar för olika smittsannolikheter i intervallet [4,7].*

# 7. Slutsats

Det som skulle hittas var tröskelvärdet för smittsannolikheten då sjukdomen utvecklas till en epidemi. Detta skulle göras för specifika värden på populationsstorleken N = 50, tillfriskningstidsintervallet [3, 9] och dödssannolikheten 0%. I figur 6.1 kan man se att konfidensintervallen för smittsannolikheterna 5.70, 5.75, 5.80 och 5.85% korsar y-värdet 1250. Vi vet alltså med stor säkerhet att tröskelvärdet för sannolikheten då sjukdomen utvecklas till en epidemi ligger i intervallet [5.70, 5.85].

# 8. Kör-instruktioner och sökväg till källkoden på KTH servrarna

Programmet är av typen Java och kräver JRE(Java Runtime Enviroment) för att köras. Programmet finns tillgängligt på kths servrar(shell.it.kth.se). Full sökväg till denna plats är följande:

* /home/a/b/abarosen/Public

Programmen består av ett paket, namngett pandemic. Detta paket/katalog innehåller två körbara klasser; Contamination och DoHundred. Båda klasserna körs i en konsol eller terminal och är alltså helt text baserade. För att köra programmet används java anropet föjlt av “paketnamnen punkt klassnamnet“. Exempel för att köra en av klasserna:

* java pandemic.Contamination

Dessa klasser kan anropas på olika sätt, med eller utan inparametrar. Källkoden finns också tillgänlig med följande sökväg:

* /home/a/b/abarosen/Public/pandemic/src

Contamination är den klass som kör simulerigen. Genom att anropa denna klass startas en simulering. Denna klass har möjligheten att använda olika inparametrar. Ifall anrop utan inparametrar sker kommer programmet ställa användarvänliga frågor för att samla in parametrarna. Det finns även ett alternativ att köra programmet helt utan inparametrar, likt ett demo, där alla parametrar antar sina skönsvärden.

Det går även att ange alla parametrarna som startargument om använderen önskar att köra programmet utan alla frågor. Inparametrarna ges i följande form S, N, minDygn, maxDygn, L.

Lista för inparametrar:

* S - står för smittsannolikheten. Detta ska ges som ett decimaltal mellan 0 till 1 (0 < S < 1).
* N - styr simulationens population. Populationen kommer till antalet vara N\*N stycken människor.
* minDygn - är antalet dagar en människa minst behöver för att tillfriskna ifrån sjukdommen.
* maxDygn - är antalet dagar en människa behöver som max för att tillfriskna ifrån sjukdommen.
* L - sjukdommens dödlighet anget i decimal form mellan 0 till 1 (0 < L < 1).

Lista för skönsvärden:

* S = 0.7
* N = 50
* minDygn = 3
* maxDygn = 9
* L = 0

Contamination med startargument kan också anropas på förenklade sätt. Dessa är med ett mindre antal startparametrar. Vid förenklade anrop används skönsvärden för de parametrar som inte blir angivna. De förenklade anropen är följande:

* ett argument: argumentet tolkas som S.
* fyra argument: Tolkar argumenten i ordningen; S, N, minDygn, maxDygn.

Exempel:

java pandemic.contamination 0.05 50 3 9

DoHundredanvänds för insamling av råa resultat och framställandet av intressanta värden kring flera körningar. Med liknande inparametrar som Contamination sätt simuleringarnas parametrar genom indata. Skilnaden är att DoHundred kör flera simuleringar efter varandra och sparar temporärt ner resultaten för varje körning. DoHundred måste anropas med startparametrar Medelvärdet och standardavvikelsen(för stickprov) räknas därefter ut och visas för användaren.

Lista för inparametrar:

* S - står för smittsannolikheten. Detta ska ges som ett decimaltal mellan 0 till 1 (0 < S < 1).
* N - styr simulationens population. Populationen kommer till antalet vara N\*N stycken människor.
* minDygn - är antalet dagar en människa minst behöver för att tillfriskna ifrån sjukdommen.
* maxDygn - är antalet dagar en människa behöver som max för att tillfriskna ifrån sjukdommen.
* L - sjukdommens dödlighet anget i decimal form mellan 0 till 1 (0 < L < 1).
* times - antalet simuleringar som ska köras.

Ett förenklat sätt att anropa programmet är att endast ange de två första parametrarna, S och N. De övriga parametrarna antar i sådana fall sina skönsvärden.

# Appendix A: Kravanalys

I tabell 1 finns en beskrivning av varje krav satt på labbuppgiften och arbetsmetodiken. I första kolumnen identifieras kravet. I andra kolumnen beskrivs kriterierna för att kravet ska vara uppfyllt. I tredje kolumnen beskriver vi hur och till vilken grad kravet är uppfyllt.

*tabell 1: kravanalys som beskriver varje krav som har satts på labbuppgiften och arbetsmetodiken och hur de har uppfyllts.*

|  |  |  |
| --- | --- | --- |
| **Namn/Beskrivning** | **Kriterier för att kravet ska vara uppfyllt** | **Om, hur och till vilken grad kravet är uppfyllt** |
| Rapportens utformning | Laborationen ska redovisas i skriftlig rapport i PDF-format. Den ska skrivas som en teknisk rapport och ska kunna förstås av läsare som inte har läst instruktionerna för laborationen. Talspråk ska undvikas i rapporten. Namn, personnummer och [mejladdress@kth.se](mailto:mejladdress@kth.se) för samtliga medlemmar i gruppen ska finnas tillgängligt i rapporten. En innehållsförteckning ska också finnas tillgänglig i rapporten. Upp till två personer kan arbeta i grupp för att lösa uppgiften. Rapporten ska lämnas in individuellt i Bilda. Om det är två medlemmar i gruppen lämnar varje gruppmedlem in en kopia av rapporten i Bilda. | Laborationen är i PDF-format. Ordningen av rubrikerna i rapporten följer grovt ordningen i lapp PM. Ordningen och rubriknamnen i labb PM är inte lik den i en typisk teknisk rapport. Rapporten struktur är därför olik den i teknisk rapport. En automatiskt genererad innehållsförteckning som täcker alla rubriker och underrubriker finns med i rapporten. Varje gruppmedlems namn, personnummer och mejladdress finns tillgänglig i gruppmedlemmarnas rapporter. |
| Utvecklings- och testmiljö | Programmet kan skrivas i valfritt språk. Programmet ska testas på någon av ICT-skolans LINUX-servrar. | Implementationen är skriven i Java. Programmet har testats på en av KTH:s Linux(Ubuntu) servrar. |
| Tolkning/beskrivning av uppgiften | Uppgiften ska tolkas och beskrivas med egna ord. Beskrivningen ska överensstämma med det som läraren avsåg med uppgiften. Beskrivningen ska vara detaljerad nog så att inga tolkningsmöjligheter finns för läsare. | Beskrivningen har gjorts genom att alla i gruppen läste genom labb PM och skrev tillsammans en utförlig tolkning av labbuppgiften i text. |
| Reflektioner över modellens egenskaper | Modellens styrkor, begränsningar och utvidgningsmöjligheter som en modell för smittspridning ska diskuteras och dokumenteras i rapporten. | Modellen jämfördes med avancerade datorspel som simulerar smittspridning och riktiga sjukdomar. |
| Beskrivning av implementationen i text | Alla enskilda komponenter d.v.s. alla klasser metoder, datastrukturer och algoritmer ska beskrivas i detalj. Beskrivningen ska vara detaljerad nog så att en läsare kan förstå alla enskilda komponenter i koden och hur de interagerar med varandra. Alla val av komponenter ska motiveras. | För att beskrivningen skulle vara detaljerad nog valde vi att beskriva varje klass separat. För varje klass beskrev vi dem globala variablerna, metoderna och algoritmerna som användes i den klassen och vad dem motsvarar i smittmodellen. Alla val av datastrukturer, algoritmer och metoder motiverades ej på grund av tidsbrist. |
| Beskrivning/Validering av implementationen med hjälp av modelleringsverktyg | Programmet ska beskrivas med UML/klassdiagram eller strukturerad programmering. Med strukturerad programmering menas en grafisk representation av programmet med till exempel flödesdiagram eller NS diagram. | Ett klassdiagram(UML) skrevs för alla java-filer i programmet Astah. |
| Indata | Användaren ska kunna sätta värden på dessa parametrar:   * Storlek på populationen N * Sannolikheten att en sjuk individ smittar en frisk granne som inte är immun S. * Gränserna för intervallet som bestämmer längden på den tid en individ är sjuk, [minDygn, maxDygn]. * Sannolikheten att en sjuk individ avlider per dygn individen är sjuk L. * Hur många individer som är sjuka initialt och var de är placerade i populationsmatrisen. | Dessa indata kan skrivas in manuellt i main metoden i klassen Contamination och DoHundred. De kan också matas in via konsolen när man Contamination klassen körs. |
| Utdata | Utdata från simuleringen:   * Antal smittade varje dygn. * Antal som avlidit varje dygn. * Antal individer som tillfrisknar varje dygn. * Antalet sjuka varje dygn. * Ackumulerat antal smittade varje dygn. * Ackumulerat antal avlidna varje dygn. | All utdata skrivs ut under en simulering för varje simulerat dygn. Utdata skrivs endast ut vid körning av Contamination klassen. |
| Validering i rapporten | I rapporten ska alla steg i arbetsprocessen från uppgiftsuppställningen till implementationen beskrivas och hur de är validerade/verifierade. Alla argument ska vara styrkta med fakta som finns tillgänglig i rapporten. | Vi antog att endast uppgiftsuppställningen och implementationen kunde valideras med hjälp utav fakta tillgänglig i rapporten. Vi beskrev därför endast de 2 arbetsstegen. |
| Validering av implementationen genom testning | En heltäckande mängd med testfall och resultatet från dem ska dokumenteras. Med en heltäckande mängd menas att testfallen ska bestå av modultester, integrationstester och systemtester. Testen ska utföras med både normala värden på in-parametrar och extremvärden. | Många testfall har beskrivits och utförts. Alla tester har dock inte utförts med normala värden på in-parametrar eftersom vi inte har kunskap om hur vi ska validera resultaten. |
| Smittmodellen | Sjukdomen sprids genom direktkontakt mellan individer. När en individ blir smittad förblir den sjuk ett slumpmässigt antal dagar i intervallet [minDygn, maxDygn]. Under varje dag en individ är sjuk finns det en viss sannolikhet att individen smittar var och en av sina åtta direkta grannar i populationsmatrisen. En individ som har smittats en dag kan inte smitta andra individer under samma dag. En sjuk individ kan inte smittas. En smittad individ kan inte dö eller smitta andra individer. Varje dag en individ är sjuk finns det en viss sannolikhet att individen avlider. En tillfrisknad individ är immun och kan inte smittas igen. | Alla krav på smittmodellen är implementerade. |
| Populationsmodellen | Populationen ska modelleras som en N\*N matris där varje element representerar en individ i populationen. | Populationen är modellerad som en tvådimensionell array där varje element innehåller ett objekt av klassen Human d.v.s. en individ. |
| Utvärdering av modellen | Utvärderingen ska baseras på en population om 50x50 individer. Initialt ska en nysmittad individ vara placeras i mitten av populationen. Sjukdomslängden ska vara rektangulärt fördelad över intervallet [3, 9] dagar. Sannolikheten L per att en smittad dör ska vara lika med 0. Det som ska hittas är tröskelvärden för smittsannolikheten så att sjukdomen utvecklas till en epidemi. Med smittsannolikheten menas sannolikheten att en individ smittar var och en av sina direkta grannar varje dygn. En epidemi definieras som att mer än 50% av alla individer smittas innan sjukdomen avklingat helt. | Experimenten som har utförts uppfyller kraven d.v.s. N = 50, [minDygn, maxDygn] = [3, 9], L = 0 och S är variabel. |
| Simuleringen | En körning av simuleringen ska pågå till dess att inga sjuka individer finns kvar i populationen. | Simuleringens slut kontrolleras via en variabel infected som alltid är lika antalet smittade och sjuka under en simulering. När infected blir lika med 0 d.v.s. när inga smittade eller sjuka finns kvar i populationen tar avslutas simuleringen. |
| Val av experiment | Alla val av experiment ska dokumenteras och motiveras. Experimenten ska vara så väl beskrivna att vem som helst kunna upprepa experimenten. | Varje enskilt experiment har beskrivits och motiverats i relation till frågeställningen. Alla in-parametrar som användes vid experimenten och vilka primtal som användes för slumptalsgeneratorn är dokumenterade. Det är därför möjligt för vem som helst med tillgång till programmet att upprepa experimenten i exakt detalj. |
| Redovisning av resultat | Resultaten ska redovisas i en graf där sannolikheten att smitta en granne(X-axeln) ska plottas mot totala antalet smittade(Y-axeln). Grafen ska plottas med konfidensintervall om mätvärdena innehåller mätfel. Läsaren ska enkelt kunna utläsa vilken tillförlitlighet och vilka felmarginaler som har använts. All information som behövs för att tolka resultaten ska finnas tillgänglig i resultatsektionen. Det ska tydligt framgå vad alla slutsatser är baserade på.  Alla grafer/tabeller ska beskrivas i texten och refereras till i texten. Varje graf/tabell ska vara numrerad och ha en beskrivande textrad. För tabeller lägger man numreringen och beskrivningen ovanför tabellen. För figurer lägger man numreringen och beskrivningen under figuren. | Resultaten är redovisade i två olika grafer med identiska axelbeteckningar. Konfidensgraden är angiven i resultatsektionen. Alla figurer refereras till i texten. |
| Statistik validitet | Körningar måste göras flera gånger med olika slumptalsfrön för att få statistiskt säkra resultat. Ett lämpligt antal slumptalsfrön är 100 st. Slumptalsfröna bör väljas som stora primtal. Värdena på slumptalsfröna ska dokumenteras i rapporten, exempelvis i en tabell. | Simuleringarna vid framtagandet av de råa resultateten görs med 100 st unika primtal. Primtalen finns dokumenterade i rapporten i form av en tabell. |
| Information om/beskrivning av hur programmet skall köras/användas | Information om de olika sätten att använda programmet på ska dokumenteras i rapporten. | Alla sätt som finns att köra klasserna Contamination och DoHundred är dokumenterat. |
| Programmets tillgänglighet | Källkoden ska laddas upp i en katalog i AFS, ett distribuerat filsystem så att Robert Rönngren kommer åt den. Användarna rron och/eller robertr ska alltså ha rl(read+list) rättighet till katalogen där källkoden samt l(list) rättighet till alla kataloger fram till katalogen med källkoden i. Vart källkoden finns i AFS ska stå i rapporten. | Källkoden har laddats upp i mappen \home\a\b\abarosen\publ  ic\pandemic\src på shell.ict.kth.se. |
| Kravanalys | Alla enskilda krav på modellen, programmet och rapporten ska identifieras och beskrivas. Hur och till vilken grad dem enskilda kraven är uppfyllda ska dokumenteras i rapporten. Kravanalysen ska göras som en checklista i tabellform med tre kolumner. Första kolumnen ska innehålla en beskrivning av varje krav. I andra kolumnen ska tolkningen av kriterierna för varje krav beskrivas. I tredje kolumnen ska det beskrivas till vilken grad varje krav är uppfyllt relativt till de tolkade kriterierna. Kravanalysen ska bifogas till rapporten som en bilaga. | Kravanalysen har skrivits enligt dem tolkade kraven. |
| Disposition | En disposition ska bifogas till rapporten. I dispositionen ska anges indelningen i stycken understycken och rubriker. Innehållet i varje stycke/rubrik ska under stycket/rubriken förklaras och motiveras. | Dispositionen som vi har skrivit är annorlunda från en typisk disposition på det sättet att den liknar en mall. |
| Källkoden | Komplett, väldokumenterad källkod ska finnas med som bilaga i rapporten. | Figurer med källkoden från alla Java-filer finns med i rapporten. Koden är väl kommenterad. Vi antar att väldokumenterad källkod är ekvivalent med väl kommenterad källkod. Vår källkod är därför väldokumenterad. |
| Rådata från körningar | All rådata som används för att utvärdera modellen ska finnas tillgänglig i rapporten i form av en tabell. Tabellen ska finnas tillgänglig i ett appendix i rapporten. | Rådata finns i två separata tabeller D.1 och D.2 som har använts för att sammanställa resultatet. |
| Reflektioner över arbetsgången | En separat sida ska bifogas till rapporten som en bilaga där individuella reflektioner över vad i arbetsgången från problemformulering, modellering och validering/verifiering som är generella ingenjörsfärdigheter som alla ingenjörer bör behärska och hur de kunskaperna kan applicera på andra områden. | Individuella reflektioner har bifogats till rapporten. Varje gruppmedlems reflektioner finns med i alla rapporter. |

# Appendix B: Disposition/Mall

**1. Introduktion**

Beskrivning av smittmodellen, uppgiften och syftet med uppgiften.

Frågor som ska besvaras i rapporten.

**2. Reflektioner kring vad som är bra, mindre bra med modellen**

Reflektioner kring modellens styrkor.

Reflektioner kring modellens svagheter och begränsningar.

Reflektioner kring modellens utvidgningsmöjligheter.

**3.1 Beskrivning av implementationen i text**

Beskrivning av vad alla klasser, metoder, algoritmer och datastrukturer motsvarar i smittmodellen och hur de interagerar med varandra.

**3.2 Beskrivning av implementationen med hjälp utav modellverktyg**

Beskrivning av implementationen med antingen UML eller strukturerad programmering.

**3.3 Validering av implementationen**

Beskrivning av enskilda testfall och resultaten från dem.

**3. Validering/Verifiering av arbetsprocessen**

Beskrivning av alla steg i arbetsprocessen d.v.s. beskrivning av uppgiftsuppställningen; modelleringen; implementationen; valideringen/verifieringen av modellen, modelleringen och implementationen; val av experiment; redovisning av resultaten; utvärdering av modellen.

Hur alla steg i arbetsprocessen är verifierade/validerade.

**5. Val av experiment**

Beskrivning av varje experiment och motivering av dem.

Tabell med slumptalsfrön som har använts under simuleringen.

**6. Resultat**

Redovisning av fakta. Graf där smittsannolikheten att smitta andra(X-axeln) plottas mot antalet smittade(Y-axeln).

**7. Slutsats**

Tolkning av resultaten d.v.s. tolkning av grafen i relation till frågeställningen.

Slutsatser, svar på frågan.

**8. Kör-instruktioner och sökväg till källkoden på KTH servrarna**

Vad för typ av applikation programmet är och vad som krävs för att köra programmet. Olika sätt att köra programmet på. Vilka in-parametrar som krävs och vilken utdata man kan förvänta sig under och efter en körning.

**Appendix A: Kravanalys**

Checklista i tabellform med tre kolumner. Kolumnerna ska innehålla en beskrivning av varje krav, tolkningen av kriterierna av varje krav och till vilken grad varje krav är uppfyllt relativt till kriterierna.

**Appendix B: Disposition**

**Appendix C: Väldokumenterad källkod**

Beskrivning av vilka filer som ingår i programmet. Indenterad källkod.

**Appendix D: Rådata från körningar**

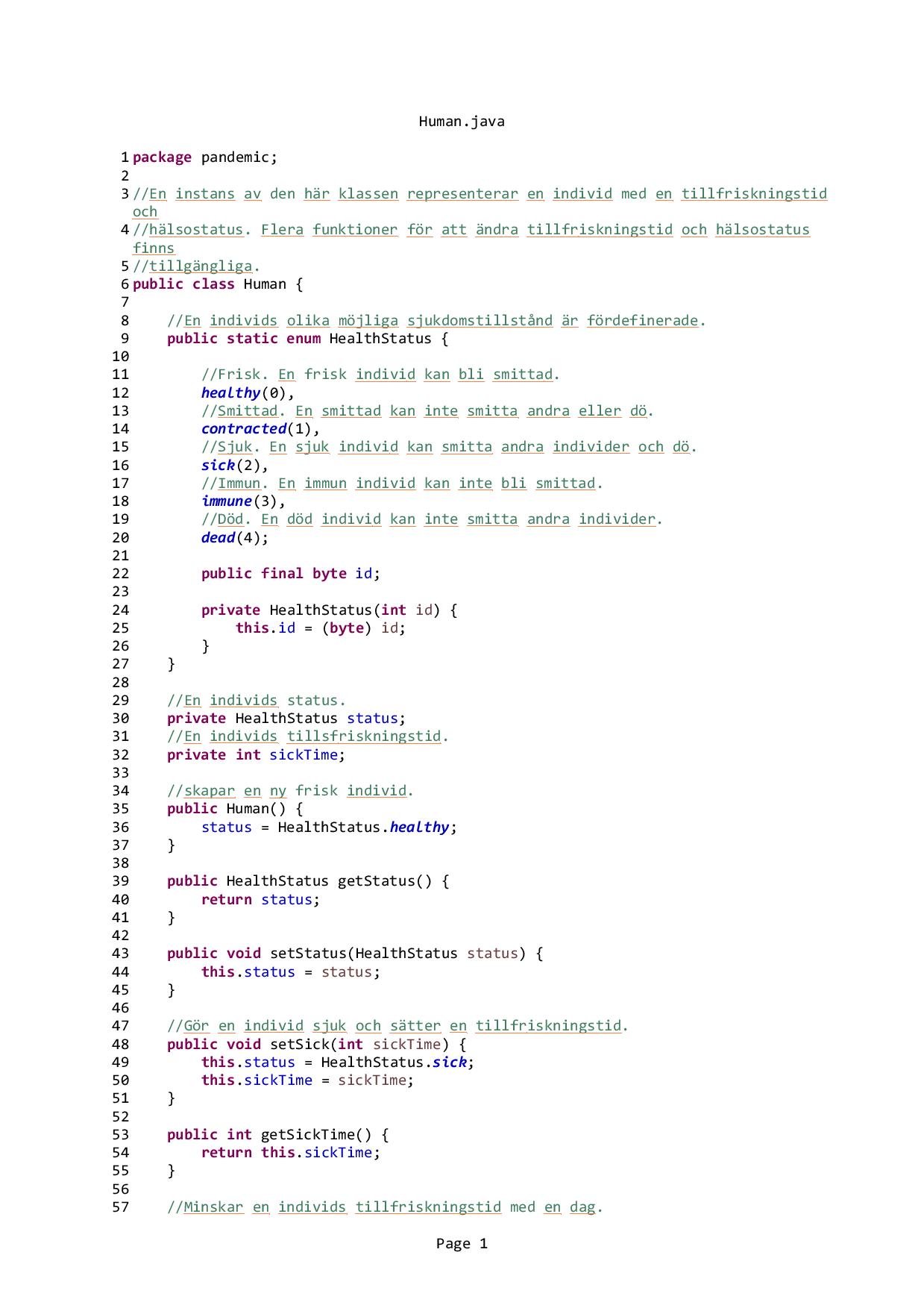
Rådata i form av en tabell.

**Appendix E: Reflektioner över arbetsgången**

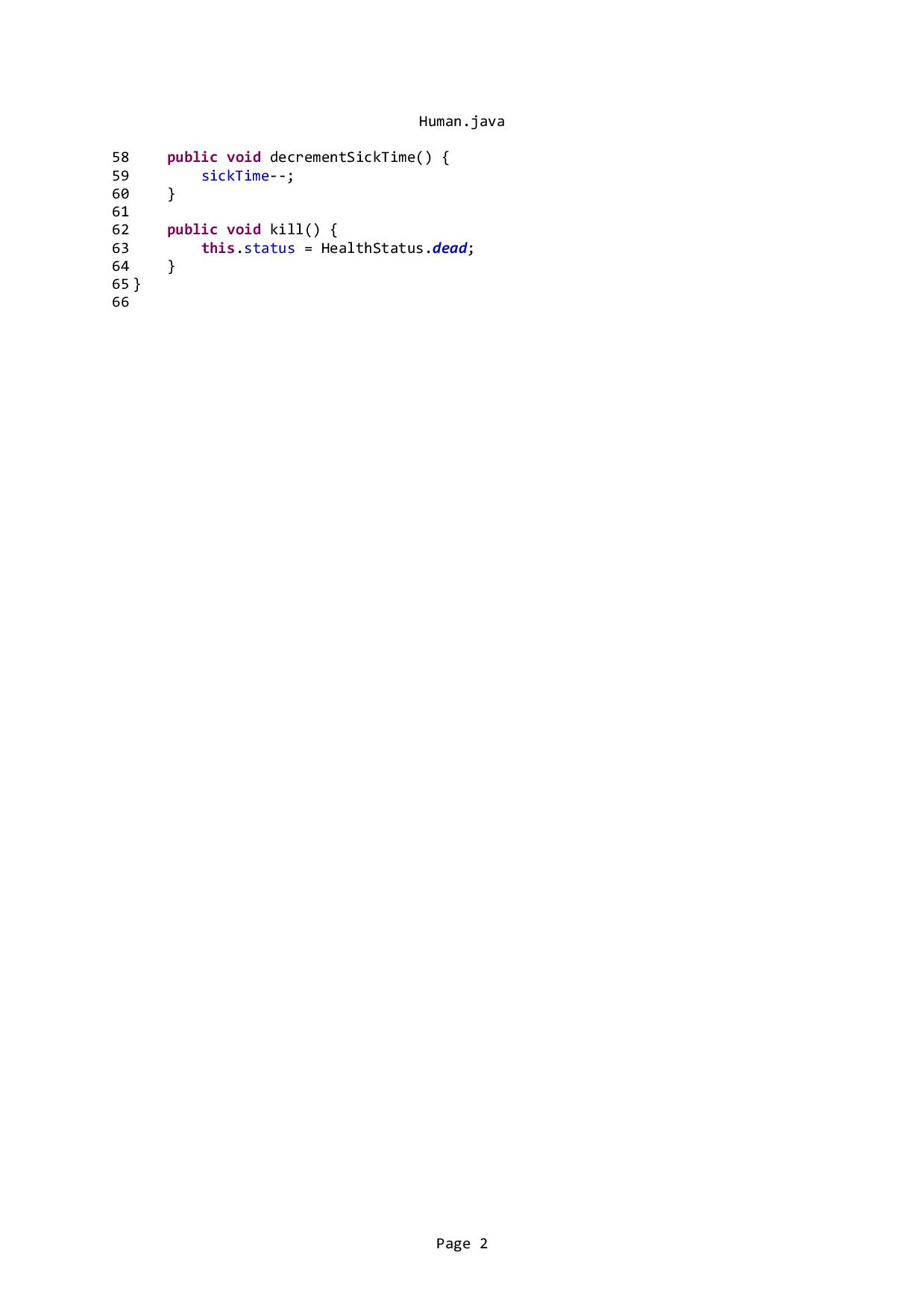
Individuella reflektioner kring vad i arbetsgången som är generella ingenjörsfärdigheter och hur de kan appliceras på andra områden.

# Appendix C: Väldokumenterad källkod

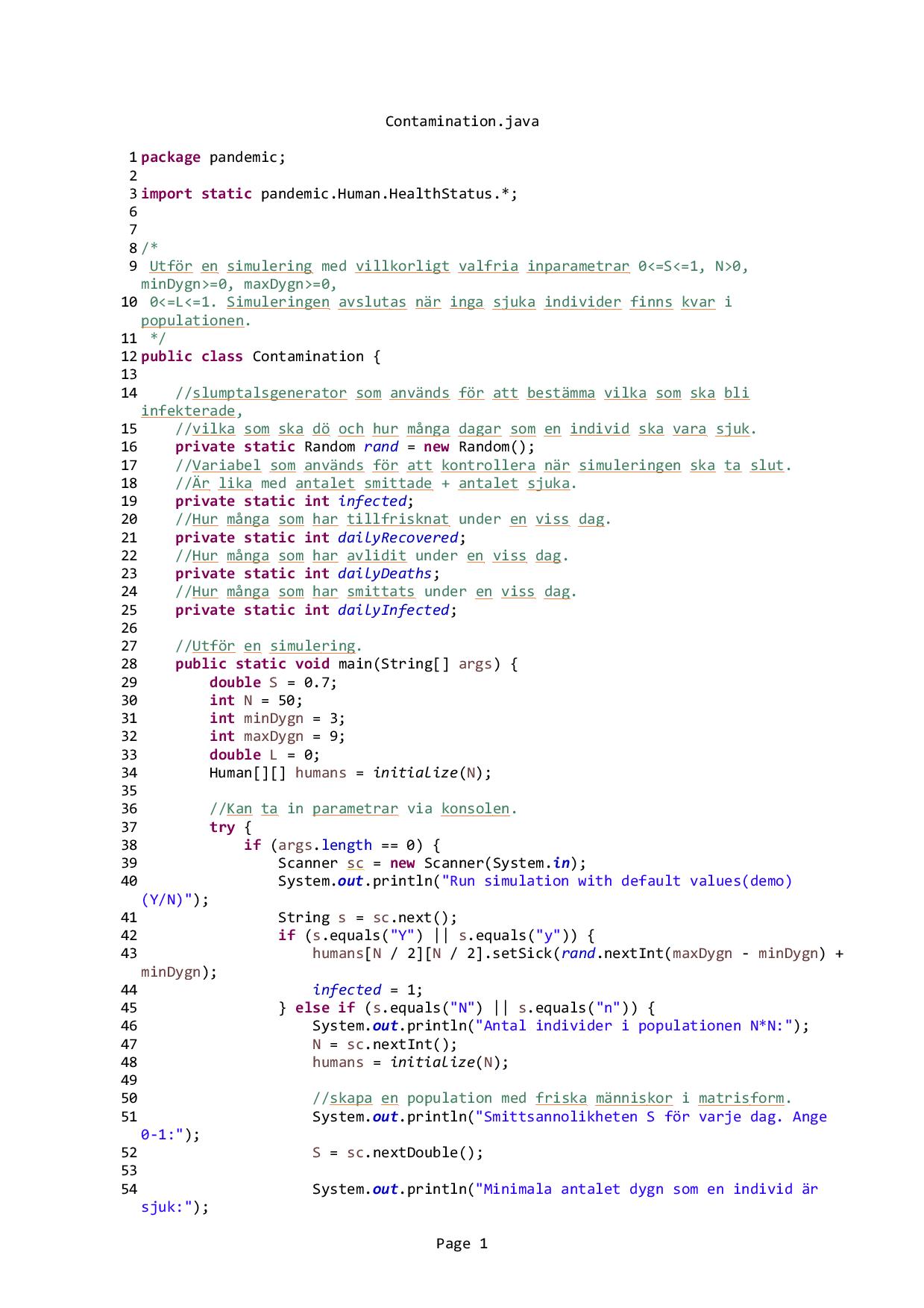
Väl kommenterad källkod illustreras i figur 4-11.



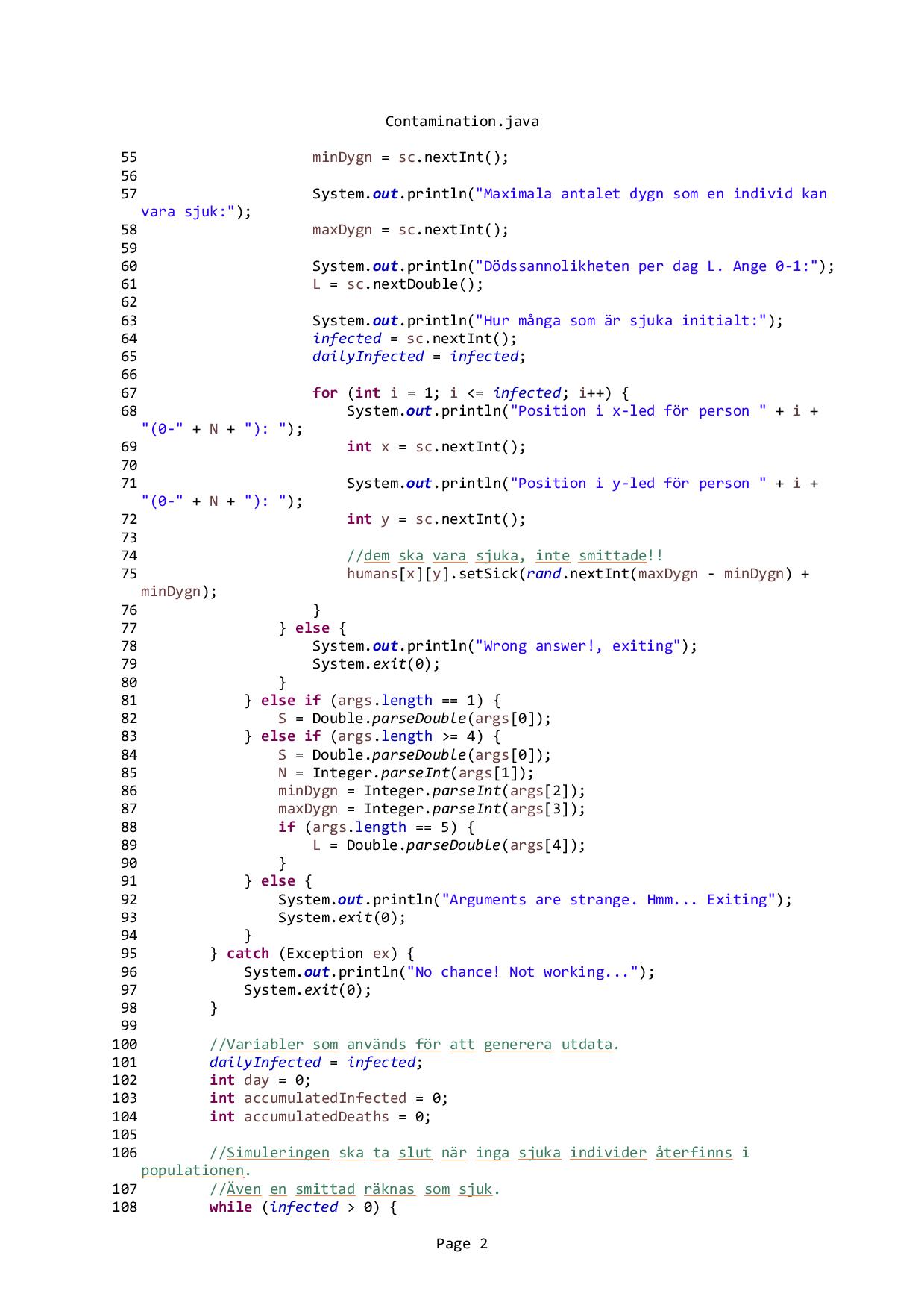
*figur 4: Källkod från klassen Human.*



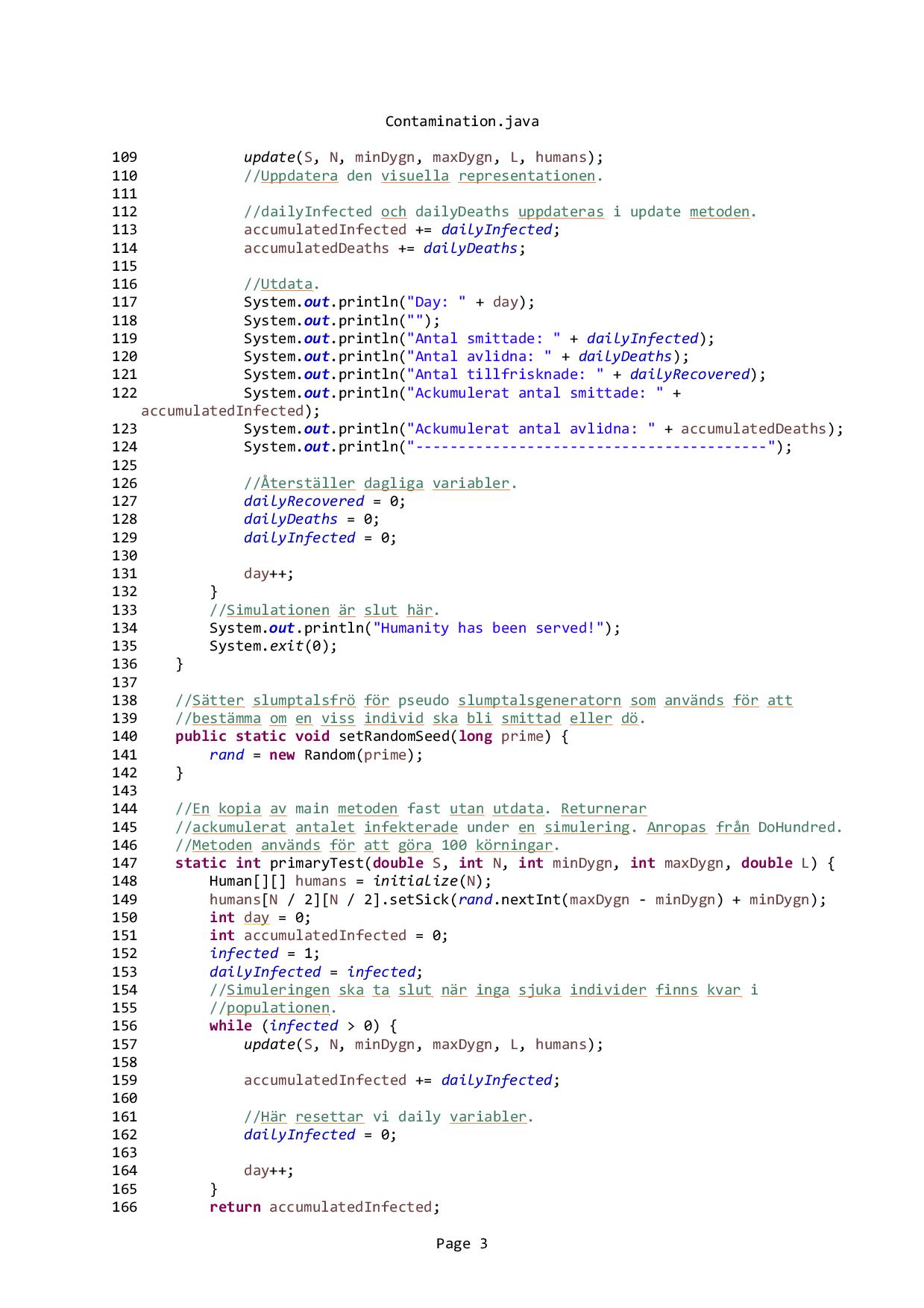
*figur 5: Källkod från klassen Human.*



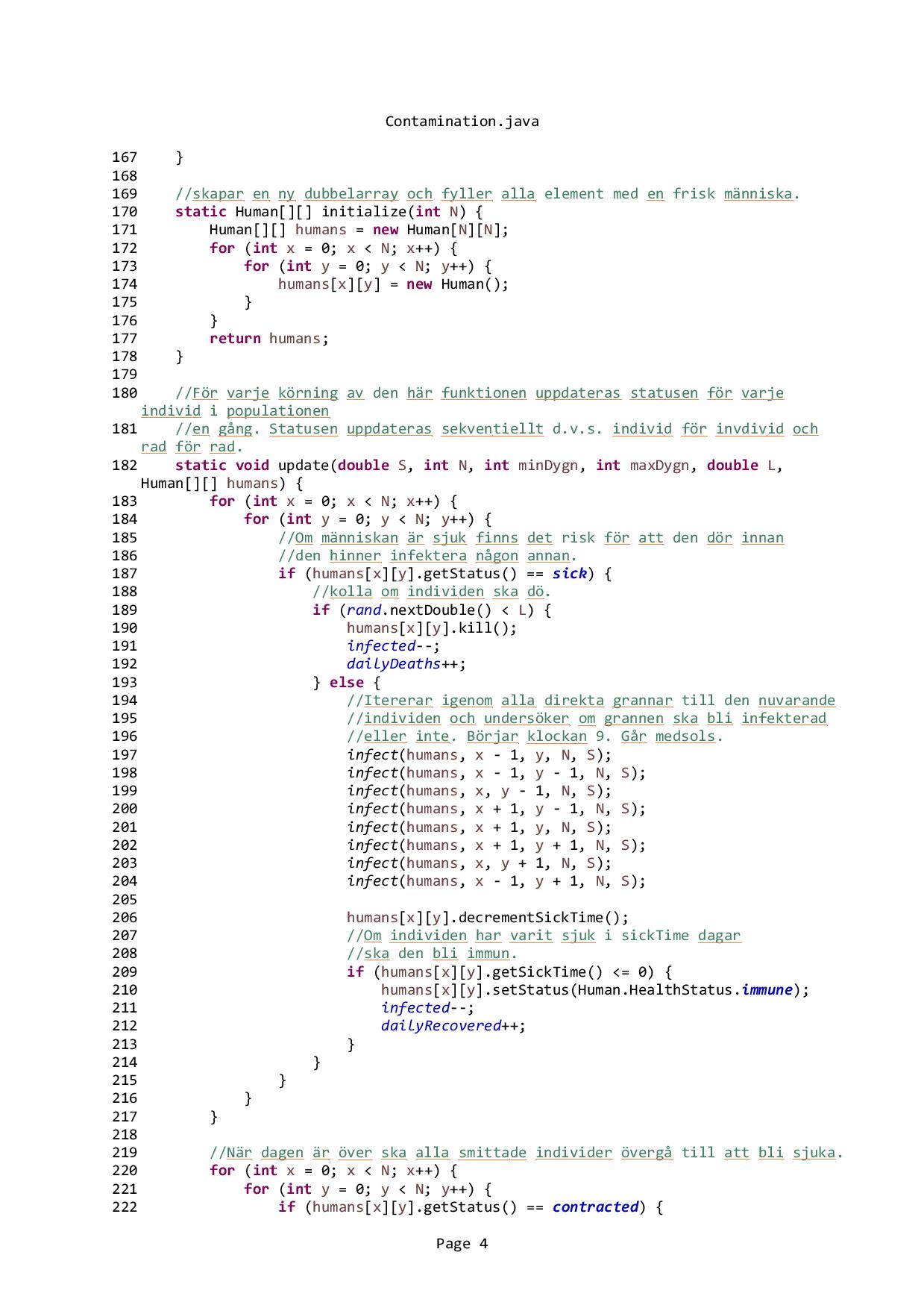
*figur 6: Källkod från klassen Contamination.*



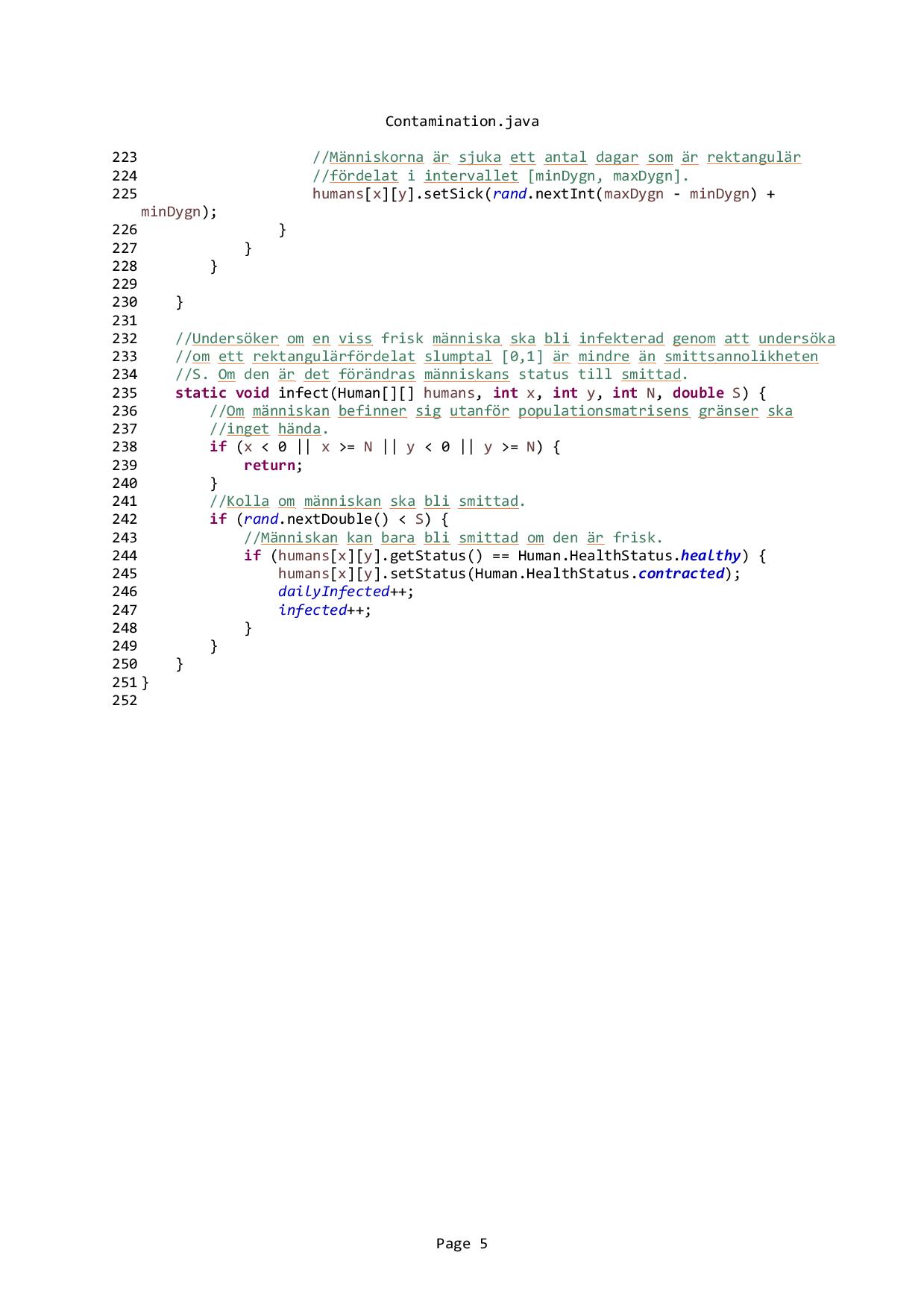
*figur 7: Källkod från klassen Contamination.*



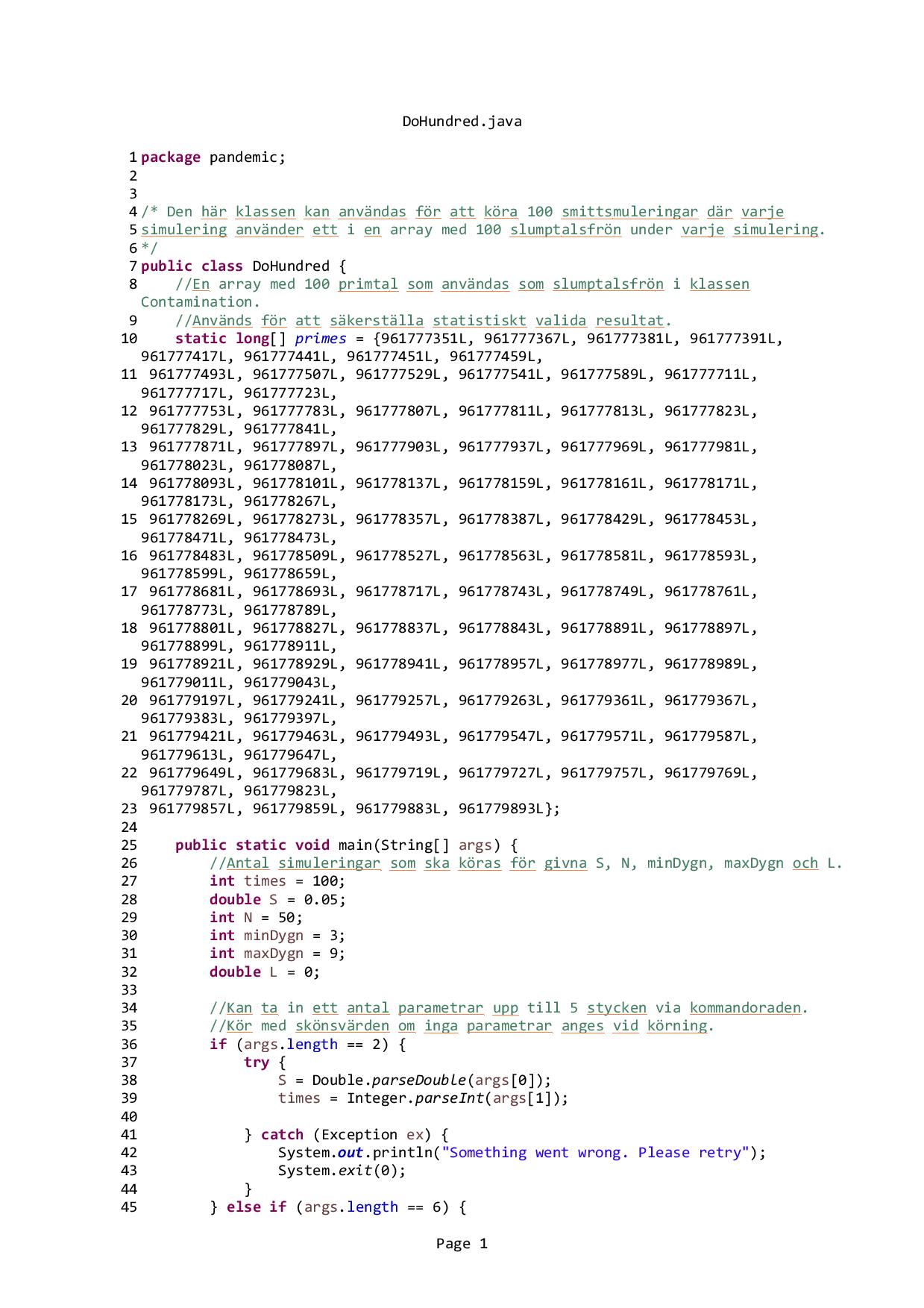
*figur 8: Källkod från klassen Contamination.*



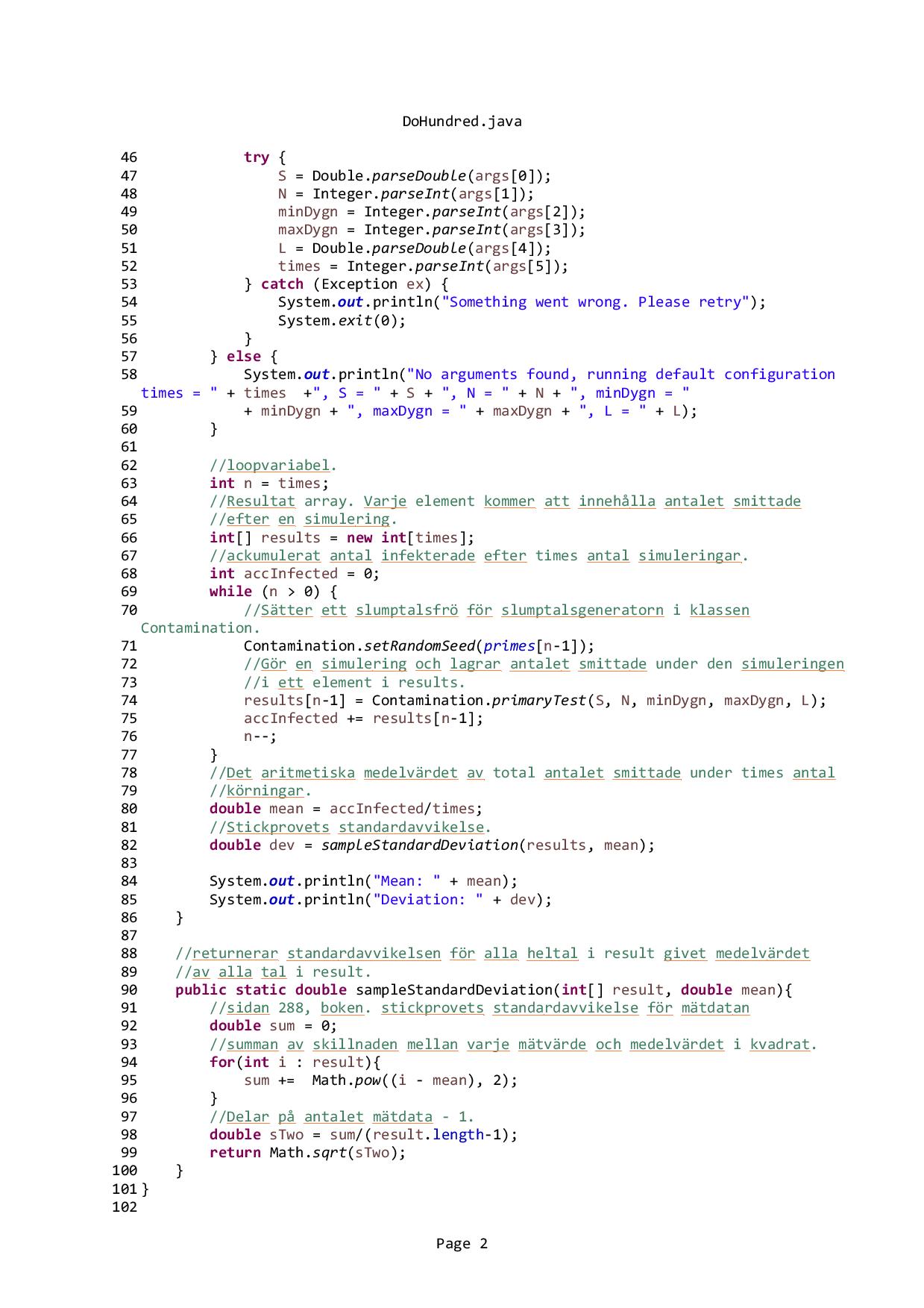
*figur 9: Källkod från klassen Contamination.*



*figur 10: Källkod från klassen Contamination.*



*figur 11: Källkod från klassen DoHundred.*



*figur C.9: Källkod från klassen DoHundred.*

# Appendix D: Slumptal som användes vid experimenten

Tabell 2 innehåller de slumptalsfrön som användes för varje enskild simulering under experimenten.

*tabell 2: Primtalen som används när klassen DoHundred körs.*

|  |  |
| --- | --- |
| 1 | 961777351 |
| 2 | 961777367 |
| 3 | 961777381 |
| 4 | 961777391 |
| 5 | 961777417 |
| 6 | 961777441 |
| 7 | 961777451 |
| 8 | 961777459 |
| 9 | 961777493 |
| 10 | 961777507 |
| 11 | 961777529 |
| 12 | 961777541 |
| 13 | 961777589 |
| 14 | 961777711 |
| 15 | 961777717 |
| 16 | 961777723 |
| 17 | 961777753 |
| 18 | 961777783 |
| 19 | 961777807 |
| 20 | 961777811 |
| 21 | 961777813 |
| 22 | 961777823 |
| 23 | 961777829 |
| 24 | 961777841 |
| 25 | 961777871 |
| 26 | 961777897 |
| 27 | 961777903 |
| 28 | 961777937 |
| 29 | 961777969 |
| 30 | 961777981 |
| 31 | 961778023 |
| 32 | 961778087 |
| 33 | 961778093 |
| 34 | 961778101 |
| 35 | 961778137 |
| 36 | 961778159 |
| 37 | 961778161 |
| 38 | 961778171 |
| 39 | 961778173 |
| 40 | 961778267 |
| 41 | 961778269 |
| 42 | 961778273 |
| 43 | 961778357 |
| 44 | 961778387 |
| 45 | 961778429 |
| 46 | 961778453 |
| 47 | 961778471 |
| 48 | 961778473 |
| 49 | 961778483 |
| 50 | 961778509 |
| 51 | 961778527 |
| 52 | 961778563 |
| 53 | 961778581 |
| 54 | 961778593 |
| 55 | 961778599 |
| 56 | 961778659 |
| 57 | 961778681 |
| 58 | 961778693 |
| 59 | 961778717 |
| 60 | 961778743 |
| 61 | 961778749 |
| 62 | 961778761 |
| 63 | 961778773 |
| 64 | 961778789 |
| 65 | 961778801 |
| 66 | 961778827 |
| 67 | 961778837 |
| 68 | 961778843 |
| 69 | 961778891 |
| 70 | 961778897 |
| 71 | 961778899 |
| 72 | 961778911 |
| 73 | 961778921 |
| 74 | 961778929 |
| 75 | 961778941 |
| 76 | 961778957 |
| 77 | 961778977 |
| 78 | 961778989 |
| 79 | 961779011 |
| 80 | 961779043 |
| 81 | 961779197 |
| 82 | 961779241 |
| 83 | 961779257 |
| 84 | 961779263 |
| 85 | 961779361 |
| 86 | 961779367 |
| 87 | 961779383 |
| 88 | 961779397 |
| 89 | 961779421 |
| 90 | 961779463 |
| 91 | 961779493 |
| 92 | 961779547 |
| 93 | 961779571 |
| 94 | 961779587 |
| 95 | 961779613 |
| 96 | 961779647 |
| 97 | 961779649 |
| 98 | 961779683 |
| 99 | 961779719 |
| 100 | 961779727 |

# Appendix E: Rådata från körningar

Tabell 3 och 4 innehåller rådata från körningar av klassen DoHundred för olika värden på smittsannolikheten S. Alla andra parametrar har haft samma värden under alla simuleringar: N = 50, [minDygn, maxDygn] = [3, 9] och L = 0. Tabell 3 har använts för att skapa grafen i figur och tabell 4 har använts för att skapa grafen i figur 3.

*tabell 3: Rådata från körningar av klassen DoHundred*

|  |  |  |
| --- | --- | --- |
| % | Aritmetiska medelvärdet för antalet smittade individer under 100 körningar. | Stickprovets standardavvikelse. |
| 0 | 1 | 0 |
| 1 | 1,63 | 1,125 |
| 2 | 2,79 | 2,724 |
| 3 | 7,07 | 11,304 |
| 4 | 31,05 | 48,365 |
| 5 | 364,86 | 413,1 |
| 6 | 1516,28 | 778,136 |
| 7 | 1995,31 | 706,264 |
| 8 | 2147,89 | 678,748 |
| 9 | 2347,67 | 414,965 |
| 10 | 2376,79 | 419,952 |
| 11 | 2394,41 | 423,017 |
| 12 | 2405,15 | 424,91 |
| 13 | 2436,61 | 349,656 |
| 14 | 2465,85 | 248,892 |
| 15 | 2465,85 | 249,206 |
| 16 | 2471,33 | 249,433 |
| 17 | 2471,9 | 249,492 |
| 18 | 2497,94 | 1,536 |
| 19 | 2498,47 | 1,251 |
| 20 | 2498,98 | 1,044 |
| 21 | 2499,29 | 0,88 |
| 22 | 2499,57 | 0,782 |
| 23 | 2499,57 | 0,671 |
| 24 | 2499,69 | 0,545 |
| 25 | 2499,77 | 0,468 |
| 26 | 2499,8 | 0,471 |
| 27 | 2499,92 | 0,273 |
| 28 | 2499,87 | 0,393 |
| 29 | 2499,96 | 0,197 |
| 30 | 2499,97 | 0,171 |

*tabell 4: Rådata från körningar av klassen DoHundred*

|  |  |  |
| --- | --- | --- |
| % | Aritmetiska medelvärdet för antalet smittade individer under 100 körningar. | Stickprovets standardavvikelse. |
| 4 | 31,05 | 48,365 |
| 4,05 | 33,87 | 57,024 |
| 4,1 | 40,46 | 63,819 |
| 4,15 | 49,89 | 81,71 |
| 4,2 | 55,26 | 86,516 |
| 4,25 | 61,07 | 96,392 |
| 4,3 | 53,53 | 86,554 |
| 4,35 | 62,83 | 97,548 |
| 4,4 | 61,92 | 90,187 |
| 4,45 | 73,49 | 109,449 |
| 4,5 | 90,69 | 148,369 |
| 4,55 | 109,97 | 177,701 |
| 4,6 | 97,48 | 151,497 |
| 4,65 | 102,67 | 153,833 |
| 4,7 | 125,36 | 205,85 |
| 4,75 | 151,98 | 225,778 |
| 4,8 | 189,1 | 267,772 |
| 4,85 | 216 | 283,108 |
| 4,9 | 251,79 | 312,241 |
| 4,95 | 305,53 | 355,99 |
| 5 | 364,86 | 413,1 |
| 5,05 | 365,59 | 429,055 |
| 5,1 | 479,05 | 509,449 |
| 5,15 | 467,05 | 482,289 |
| 5,2 | 562,68 | 521,531 |
| 5,25 | 616,94 | 511,138 |
| 5,3 | 643,91 | 556,039 |
| 5,35 | 735,27 | 602,185 |
| 5,4 | 740,92 | 641,78 |
| 5,45 | 887,55 | 671,231 |
| 5,5 | 894,43 | 644,637 |
| 5,55 | 996,51 | 711,336 |
| 5,6 | 1064,89 | 733,352 |
| 5,65 | 1034,83 | 727,323 |
| 5,7 | 1178,99 | 743,95 |
| 5,75 | 1284,54 | 739,266 |
| 5,8 | 1296,88 | 756,087 |
| 5,85 | 1364,67 | 759,761 |
| 5,9 | 1408,1 | 759,974 |
| 5,95 | 1442,84 | 758,436 |
| 6 | 1516,28 | 778,137 |
| 6,05 | 1544,85 | 753,189 |
| 6,1 | 1612,93 | 752,273 |
| 6,15 | 1662,76 | 743,521 |
| 6,2 | 1688,17 | 731,643 |
| 6,25 | 1722,51 | 725,944 |
| 6,3 | 1725,49 | 745,207 |
| 6,35 | 1757,05 | 751,989 |
| 6,4 | 1788,67 | 757,387 |
| 6,45 | 1769,37 | 782,993 |
| 6,5 | 1777,72 | 782,193 |
| 6,55 | 1802,98 | 792,871 |
| 6,6 | 1806,9 | 796,247 |
| 6,65 | 1837,09 | 780,902 |
| 6,7 | 1899,09 | 738,896 |
| 6,75 | 1932,68 | 719,664 |
| 6,8 | 1928,48 | 719,621 |
| 6,85 | 1956,05 | 727,328 |
| 6,9 | 1962,41 | 728,888 |
| 6,95 | 1972,45 | 732,314 |
| 7 | 1995,31 | 706,264 |

# Appendix F: Individuella reflektioner över arbetsgången

## F.1 Alexander Barosens reflektioner över arbetsgången

Något en ingenjör borde ha är breda erfarenheter för att enklare kunna sätta sig in i ett problem, även om problemet kanske inte är specifikt definierat i ett visst område, så ska ingenjör fortfarande kunna lösa uppgiften. Ett bra sätt att bli bättre att sätta sig in i nya problem kan vara att studera modeller och ha modelltänkandet när man försöker lösa en uppgift. Det hjälper till att skapa en uppfattning om vilka gränser och mål en uppgift kan ha.

Det är också viktigt att kunna redovisa och presentera resultat på ett bra sätt. Detta är viktigt för att andra ska kunna ta del utav en utförd process. Därav blir ett problem eller lösning en lärdom för andra ingenjörer. Detta är också mycket gynnsamt i senare utveckling, så att man slipper upprepa tidigare misstag eller för att förkorta arbetstiden. Därför är det viktigt att redovisningen sker på ett tydligt sätt där man kan förstå problemet.

## F.2 Sadok Habibi Dalins reflektioner över arbetsgången

Metoderna vi har använt i den här uppgiften för att validera implementationen och modelleringen kan appliceras på nästan alla områden där hypotesprövning görs. Att kunna presentera sin arbetsmetodik och visa hur den är validerad/verifierad är viktigt inom all typ av forskning. Ett korrekt resultat är utan värde om inte metoden för att framställa resultat är korrekt. Rapporter kan användas för detta syfte. Testningsmetodiken som har använts här kan appliceras på all mjukvaruutveckling.