Ohjelmointi 5: Miinaharava

Taustaa

Tämänkertainen kotitehtävä ei enää liity aikaisemmilla kierroksilla kehittelemäämme seikkailupeliin. Nyt toteutamme graafisen miinaharavapelin. Opimme käyttämään Javan Swing-luokkakirjastoa, jonka avulla teemme pelin käyttöliittymän.

Suorakulmainen pelikenttä koostuu ruuduista, joista osaan on piilotettu miina – miinojen sijaintia ei kuitenkaan tunneta pelin alussa. Niissä ruuduissa, joissa ei ole miinaa, on piilossa vihjenumero. Se kertoo, montako miinaa kyseistä ruutua ympäröivissä kahdeksassa naapuriruudussa on yhteensä. Mikäli vihjenumero olisi nolla, ruutu näytetään tyhjänä.

Mahdollinen vihjenumero (tai pahimmassa tapauksessa miina) paljastuu vasta pelaajan naksauttaessa ruutua hiiren vasemmalla napilla. Ruudut, joissa vihjeiden perusteella tiedetään olevan miina, voidaan merkitä varoituslipulla klikkaamalla hiiren oikeaa painiketta ruudun päällä. Peli on voitettu, kun kaikkien miinojen paikat on selvitetty. Tappio seuraa yhdenkin miinan kaivamisesta esiin.

Jos peli ei ole tuttu, voit ottaa tuntumaa tämänkertaiseen tehtäväkenttään esimerkiksi XEmacsissa valitsemalla valikosta Tools -> Games -> Mine Game. Älä kuitenkaan unohdu XEmacs-version pariin liian pitkäksi aikaa – muista, että tämän kierroksen lopussa sinulla on paljon parempi peli pelattavanasi! :)

Osatehtäviä on tällä kertaa paljon, mutta osatehtäviä 5 ja 6 lukuun ottamatta ne ovat varsin pieniä. Ne kannattaa tehdä järjestyksessä. Kannattaa myös lukea jokainen osatehtävä kokonaan, ennen kun ryhtyy ohjelmoimaan sitä. Käännä ohjelmasi jokaisen osatehtävän jälkeen ja testaa, että uusimmat lisäykset myös toimivat oikein.

Tehtävässä tavoiteltava lopputulos – toimiva peli – voi näyttää vaikkapa tältä:


Miinaharava
Peli
Peli on kesken.


Miinaharava
Peli
KABOOM! Hävisit!

Tarvittavat taidot

Jotta saisit tehtävän tehtyä, on sinun hallittava edellisillä kotitehtäväkierroksilla vaadittujen taitojen lisäksi ainakin seuraavat Swing-luokkakirjastoon liittyvät taidot:

- Swingin peruskomponenttien (JFrame, JPanel, JLabel, JButton) muodostaminen ja käyttö.
- Valikkorivin rakentaminen.
- Asettelijoiden (BorderLayout, GridBagLayout) käyttö.
- Komponenttien ominaisuuksien (väri, koko, reunus, teksti) muuttaminen.
- Tapahtumankuuntelijoiden (ActionListener, MouseListener) käyttö.

---

1. osatehtävä: Kehysluokka `Miinapeli`

Aloitetaan laatimalla pääohjelmaluokka `Miinapeli`, joka perii luokan `JFrame`. Ikkunan sisällön asettelijana käytetään `BorderLayout`:ia. Sijoita ikkunan keskelle suuri harmaa `JPanel`-paneeli, johon ei tarvitse laittaa vielä mitään. Aseta tälle paneelille suosituskoko (preferredSize), jota se pyrkii noudattamaan – esimerkiksi 400*300 pikseliä.

Ikkunan alareunaan laitetaan toinen paneeli, jonka taustavärin voit valita vapaasti (kunhan se näyttää hyvältä :). Tähän paneeliin asetetaan `JLabel`-komponentti, joka kertoo pelin tilan. Toistaiseksi komponentissa voi olla vaikkapa teksti "Peli on kesken". Voit asettaa paneelille reunukset ja muotoilla sen ulkoasun haluamaksesi. Tähän ei kuitenkaan kannata tässä vaiheessa käyttää liikaa aikaa, vaan voit palata hiomaan ulkoasua, kun peli on muuten valmis.

Lisäksi ikkunaan laitetaan valikkorivi ja siihen yksi valikko nimeltä "Peli". Tähän valikkoon lisätään rivit "Aloita alusta" ja "Lopeta", jotka erotetaan toisistaan vaakaviivalla (separator). Toistaiseksi komponentteihin ei tarvitse liittää mitään kuuntelijoita. Aseta ikkunallesi myös otsikko.

Laadi lopuksi luokalle pääohjelmametodi, joka luo uuden `Miinapeli`-olion ja asettaa sen näkyviin. Ikkuna ei todennäköisesti aukea aivan oikean kokoisena – ongelman ratkaisemiseksi kannattaa tutustua `JFrame`-luokan (alun perin luokasta `java.awt.Window` perittyyn) metodiin `pack()`. Lisäksi haluaisimme, että ikkuna aukeaisi vasemman ylänurkan sijaan keskelle ruutua. Tämä saavutetaan puolestaan metodilla `setLocationRelativeTo(Component c)`, jonka `JFrame` on niin ikään perinyt luokalta `Window`.

Huomaa, että sulkiessasi ikkunan itse ohjelma jää vielä taustalle pyörimään. Voit päättää ohjelman suorituksen näppäinkomennolla `Ctrl+c` tai katsoa suoraan seuraavan osatehtävän viimeisestä kohdasta pysyvämmän ratkaisun ongelmaan.

2. osatehtävä: Tapahtumien kuuntelijat

Seuraavaksi rakennetaan ohjelmaan hieman vuorovaikutteisuutta liittämällä ylävalikon painikkeisiin `ActionListener`-rajapinnan toteuttavat kuuntelijat, jotka reagoivat käyttäjän tekemisiin. Kuuntelijana voidaan käyttää omaa pelkästään kuunteluun erikoistunutta luokkaansa, tai sellaisena voi toimia mikä tahansa muukin luokka toteuttamalla kuuntelijarajapinnan. Kuuntelijaksi kannattaa valita olio, jolla on mahdollisimman vaivaton pääsy kuuntelusta seuraaviin toimintoihin.

Tässä kuuntelijat toteutetaan harjoituksen vuoksi ns. sisäluokkien (engl. nested class tai inner class) avulla. Sisäluokat ovat toisen luokan sisällä määriteltyjä, yleensä pieniä apuluokkia, jotka mm. näkevät suoraan kaikki isäntäluokkansa attribuutit. Nyt laadittavat sisäluokat toteuttavat rajapinnan `ActionListener`. Sisäluokkien käyttämisestä kuuntelijoina on esimerkkejä Kalakirjan luvussa 14.4 ja Sunin Java-tutoriaalissa.

- Lopeta-painikkeeseen liitetään kuuntelija, joka osaa pyydettäessä sulkea ohjelman. Ohjelma lopetetaan metodikutsulla `System.exit(0)`.

- Liitä kuuntelija myös ylävalikon Aloita alusta -painikkeeseen. Toistaiseksi meillä ei ole peliä, jonka voisimme aloittaa alusta, joten painikkeelle laitetaan tässä vaiheessa vain muuttamaan alapaneelin tekstiä. Tekstiksi voi vaihtua vaikkapa "Et vaan osaa" tai "Vain rekisteröidyssä versiossa".

- Lisäksi itse ikkunalle pitää määritellä toiminto, joka sulkee ohjelman, kun oikean yläkulman X-ikonia painetaan. Muuten ikkuna kyllä katoaa näkyvistä, mutta taustalla pyörivä ohjelma jatkaa toimintaansa. Ikkunalle voitaisiin laatia oma kuuntelija (`WindowListener`), mutta helpommalla selviät, kun tutkit, miten `JFrame`:n metodi `setDefaultCloseOperation` toimii.

3. osatehtävä: Luokka `Pelipaneeli`

Seuraavaksi luodaan paneeli, johon sijoitetaan miinakenttää kuvaava nappiruudukko. Pelipaneeli perii luokan `JPanel`, ja sen sisällön asettelijana

käytetään `GridBagLayout`:ia. Huomaathan, että tämän asettelijan käyttö vaatii paikkatietojen syöttämisen `GridBagConstraints`-olion avulla.

Paneeliin ladottavilla napeilla on kullakin kaksi koordinaattia (x, y) niin, että x-koordinaatti kasvaa vasemmalta oikealle ja y-koordinaatti ylhäältä alas. Molemmat koordinaatit alkavat nollasta. Pelipaneelille tehdään sopiva tietorakenne, johon luodut napit tallennetaan. Napit toteutetaan aluksi `JButton`-luokan avulla tavallisina harmaina painikkeina. Niillekin on syytä asettaa oletusarvoinen koko, esimerkiksi 25*25 pikseliä.

Pelipaneelille tehdään konstruktori, jolle annetaan tässä vaiheessa kaksi parametria – ruudukon leveys ja korkeus (montako nappia ruudukossa on vaaka- ja pystysuunnassa):

```
public Pelipaneeli(int leveys, int korkeus)
```

Leveyden ja korkeuden on molempien oltava positiivisia. Konstruktorissa luodaan kaikki napit ja asetetaan ne paikoilleen `GridBagLayout`:ia käyttämällä.

Testaa aikaansaannostasi luomalla `Miinapeli`-luokassa uusi `Pelipaneeli` sopivilla konstruktorin parametreilla (kannattaa kokeilla useita eri arvoja) ja lisäämällä se ikkunan keskelle, jossa aiemmin oli tyhjä harmaa paneeli.

4. osatehtävä: Hiiren kuunteleminen

Miinaharavassa käytetään hiiren molempia nappeja (vasenta ja oikeaa). Edellä käyttämämme `ActionListener` kuuntelee kuitenkin vain vasemman painikkeen naksautuksia, joten se ei riitä tähän tarkoitukseen. Asetammekin pelipaneelin napeille kuuntelijan, joka toteuttaa rajapinnan `MouseListener`. (Huomaa, että `BUTTON1` viittaa hiiren vasemmanpuoleiseen ja `BUTTON3` oikeanpuoleiseen nappiin.)

Sen sijaan, että jokaiselle napille luotaisiin oma kuuntelijaolionsa, käytämme tässä kuuntelijana `Pelipaneeli`-luokkaa. Laita se siis toteuttamaan rajapinta `MouseListener`. Huomaa, että rajapinta vaatii luokkaan kaikkiaan viisi erilaista metodia, mutta tarvitsemme oikeasti niistä vain yhtä. Loput neljä voi jättää tyhjiksi, kunhan ne kuitenkin ovat luokassa mukana.

Laadi kuuntelija tässä vaiheessa sellaiseksi, että se muuttaa vasemmalla painikkeella naksautetun `JButton`:in siniseksi ja oikealla painikkeella naksautetun vihreäksi. Testaa, että napit käyttäytyvät halutulla tavalla.

5. osatehtävä: Miinaharavan pelilogiikka

Seuraavaksi rakennamme pelimme pohjaksi sen varsinaisen logiikkakoneiston, joka pitää yllä tilannekuvaa pelikentästä: siitä, missä ruuduissa piileksii miinoja, mitkä ruudut on jo aukaistu ja mitkä on merkitty varoituslipulla. Laadimme pelilogiikan omaan luokkaansa, joka ei ole riippuvainen mistään tietystä käyttöliittymätoteutuksesta, vaan sitä voisi sopivan julkisen rajapinnan (eli sopivien metodien) kautta käyttää monella erilaisella graafisella tai ei-graafisella käyttöliittymällä. *Yleensä*, ellei kyse ole hyvin yksinkertaisesta ohjelmasta, on hyvä ajatus erottaa sovelluslogiikka ja käyttöliittymä toisistaan – tällöin toiseen on helppo tehdä muutoksia ilman, että toista joudutaan merkittävästi sopeuttamaan muutoksiin.

Miinapelimme looginen ydin on luokka nimeltä `Peliruudukko`. Luokan toteutuksen yksityiskohdat saat suunnitella itse – vaatimuksena kuitenkin, että se toteuttaa Nopasta löytyvästä dokumentaatiossa määritellyn julkisen rajapinnan. Tutustu huolella siihen, mitä eri metodien ja julkisten vakioiden kuvauksissa sanotaan.

Vihjeitä luokan toteutukseen:

- Ruudukko kannattaa esittää kaksiulotteisena taulukkona. Yksi vaihtoehto on laatia apuluokka (esimerkiksi `Peliruutu`), joka kuvaa ruudukon yksittäistä ruutua ja pitää tallessa ruudun kannalta olennaiset tiedot: onko ruutu jo avattu, onko ruutu merkitty lipulla ja onko ruutuun kätketty miina.
- Toinen, vähemmän oliolähtöinen mutta ehkä hieman suoraviivaisempi tapa on kuvata näitä tietoja suoraan `Peliruudukko`-luokassa kolmella eri kaksiulotteisella `boolean`-taulukolla.
- Haastavin osa tätä osatehtävää on todennäköisesti ruudun vihjenumeron selvittäminen. Sitä varten joudut käymään läpi ruudun kaikki mahdolliset naapuriruudut (joita voi siis olla maksimissaan kahdeksan, reuna- ja nurkkaruuduilla kuitenkin vähemmän) ja tutkimaan, moniko niistä kätkee sisäänsä miinan. Tämän voi tehdä hyvin monella eri tavalla – valitse sellainen, joka tuntuu sinusta selkeimmältä. Järkeilemällä ja miettimällä voit keksiä, miten pääset tarvittaviin tietoihin käsiksi ilman, että käyt tekemässä ”rumaa koodia” esim. tekemällä attribuuteista tai metodeista staattisia tai julkisia aivan turhaan. Muista kuitenkin kommentoida oma ratkaisusi, päädyit mihin tahansa.
- Dokumentaatiossa mainittuja poikkeuksia ei tarvitse erikseen heittää tai edes määritellä heitettäväksi `throws`-avainsanalla. Ne lentävät kyllä itsestään, mikäli ongelmallinen tilanne tulee vastaan. Koordinaatteja ei

siis tarvitse näissä metodissa tarkistaa, vaan jos parametrit olivat kelvottomat, metodi saa heittää asianmukaisen poikkeuksen – valmiissa pelissä metodia ei koskaan pitäisi päästä kutsumaan muilla kuin "hyvillä" koordinaateilla, joten ohjelman kaatuminen mahdollisessa virhetilanteessa todennäköisesti vain auttaa bugien löytämistä.

- Voit testata ruudukkosi toimintaa Nopasta löytyvän `Miinatesti`-luokan avulla. Se käyttää ruudukkoa apuna yksinkertaisessa merkkipohjaisessa miinaharavapelissä, jota pelataan syöttämällä aina sen ruudun koordinaatit, jonka seuraavaksi haluat avata. Liputus, pelin voittaminen ja muut korkeamman tason featuret puuttuvat tästä pelistä, mutta sillä saanee hyvän käsityksen siitä, toimiiko peliruudukkosi siten kuin on tarkoitus.

Kun peliruudukkosi on valmis, on aika kytkeä se edellä luomaamme graafiseen pelipaneeliin. Muuta `Pelipaneeli`-luokan konstruktoria niin, että leveyden ja korkeuden sijaan se ottaa nyt parametrinaan vain `Peliruudukko`-olion ja asettaa oman leveytensä ja korkeutensa ruudukon mittojen mukaan. Laita lisäksi pelipaneeli värittämään testausmielessä mustiksi kaikki ne napit, joiden paikalla peliruudukossa on miina. Kokeile asettaa peliruudukon luontimetodille erilaisia parametreja – leveyksiä, korkeuksia ja miinamääriä – ja katso, minkälaisia näkymiä saat aikaan.

6. osatehtävä: Luokka `Ruutunappi` ja graafinen miinaharava

Tässä osatehtävässä korvataan pelipaneelin `JButton`:it hieman erikoistuneemmilla napeilla, joilla on Miinaharava-pelin tarpeisiin laadittuja toimintoja. Napit esittävät edellisen osatehtävän ruudut graafisesti, joten myös ne on voitava avata tai liputtaa. Tätä varten laaditaan luokka `Ruutunappi`, joka perii luokan `JButton`. Ruutunapin on tiedettävä omat koordinaattinsa pelipaneelissa, ja näitä tietoja on myös voitava kysyä napilta.

Pelin alussa kaikki napit ovat harmaita. Naksautettaessa napin väri ei enää muutu siniseksi tai vihreäksi. Sen sijaan nappi käyttäytyy tästä eteenpäin hieman miinaharavamaisemmin:

- Kun nappia klikataan hiiren vasemmalla nappulalla, välitetään `Peliruudukko`-oliolle tieto kyseisen ruudun avaamisesta – ruutu avautuu, mikäli se ei ollut liputettu tai jo entuudestaan avattu. Avaaminen näytetään myös visuaalisesti niin, että napin tausta muuttuu valkoiseksi ja napin reunus (engl. border) vaihtuu toisenlaiseksi. Reunusten avulla esitetään se, että napit ovat alussa hieman koholla alustastaan ja esiin kaivettuina näyttävät painuvat

alemmas alustan tasoon – katso mallia vaikkapa tehtävänannon alussa olevista kuvista. Reunusten käytöstä on lisää tietoa Sunin Border-tutoriaalissa. Lisäksi avatussa napissa näytetään ruudusta paljastunut mahdollinen vihjenumero tai miina.

- Kun nappia klikataan hiiren oikealla painikkeella, välitetään `Peliruudukko`-oliolle tieto kyseisen ruudun merkkaamisesta lipulla. Liputus esitetään myös visuaalisesti muuttamalla nappi oranssiksi ja lisäämällä sen keskelle musta piste. Uusi oikean hiirennapin painallus merkityn napin päällä ottaa vastaavasti lipun pois eli palauttaa napin harmaaksi ja poistaa pistemerkin.

Toteuta näiden ohjeiden mukaan napille seuraavat metodit, jotka muuttavat nappien ulkonäköä pelitilanteeseen sopivaksi:

```
public void naytaMiina()
```

Avaa napin eli muuttaa sen reunuksen ja taustavärin siten kuin osatehtävän alussa kuvattiin. Lisäksi nappi näytetään miinana laittamalla siihen miinan kuva, joka on tässä vaiheessa @-merkki.

```
public void naytaVihje(int vihjenumero)
```

Myös tämä metodi avaa napin ja muuttaa sen ulkoasua asianmukaisesti. Lisäksi nappi näyttää vihjenumeronsa, paitsi jos vihjenumero on nolla, jolloin nappi jää ilman tekstiä. Tee eri vihjenumeroista erivärisiä. Hyvin valituilla väreillä saat pelistä paljon selkeämmän kuin pelkillä yhden värisillä vihjeillä.

```
public void naytaLippu(boolean lippu)
```

Asettaa napille lipun (eli edellä kuvatun pisteen ja oranssin värin) tai ottaa sen pois, parametrin arvosta riippuen. Jos napilla on jo lippu, tämä metodi parametrilla `true` ei tee mitään näkyvää (eli vain asettaa napin ulkoasun sellaiseksi kuin se jo oli), vastaavasti ei myöskään parametrilla `false` mikäli napilla ei sillä hetkellä ole lippua.

Laita nyt hiiren kuuntelija reagoimaan ruutunappien klikkaamiseen sopivalla tavalla. Ensin se tutkii `Peliruudukko`-oliolta kyseisen ruudun tilan varmistaakseen, että klikkaukseen ylipäänsä kuuluu reagoida. Tämän jälkeen, mikäli aiheellista, kuuntelija kutsuu `Peliruudukko`-olion metodia, jolla kyseinen ruutu avataan tai liputetaan. Sitten muutetaan vielä klikatun ruutunapin ulkoasu asianmukaiseksi, sen perusteella, mitä peliruudukon ruudunavausmetodi palautti. Kaikki yhden ruudun avaamiseen liittyvät toimenpiteet kannattanee sijoittaa myös `Pelipaneeli`-luokassa omaan

apumetodiinsa, joka saa parametreinaan esimerkiksi avattavan ruudun koordinaatit.

Huomaa myös, että paljastettu nappi ei siis enää reagoi naksautukseen hiiren kummallakaan painikkeella (ymmärrettävää – sitä ei oikein voi enää kaivaa esille eikä merkitäkään). Jos taas nappi on merkittynä, vasemmalla hiirennapilla klikkailu ei vaikuta siihen. Näin vältytään klikkaamasta vahingossa auki nappeja, jotka "varmasti" tiedetään miinoiksi.

Kun pelaaja klikkaa esiin napin, jonka vihjenumero olisi nolla, kaivetaan tällöin esille myös kaikki tuon napin vielä avaamattomat naapurit (huomaa, ettei riski osua tällöin vahingossa miinaan ole suuren suuri..). Tätä varten `Pelipaneeli`-luokkaan voidaan laatia vielä esimerkiksi seuraavanlainen apumetodi:

```
private java.util.List<Ruutunappi> poimiNaapurit(int x, int y)
```

> Se palauttaa listan kaikista annetuissa koordinaateissa sijaitsevan napin *naapureista* (enintään kahdeksan kappaletta). Naapurien selvittämiseksi on jälleen olemassa monenlaisia ratkaisuja, jotka muistuttavat hyvin paljon vihjenumeron laskemista edellisessä osatehtävässä.

Kytke tämä metodi nappien naksuttelulogiikkaan niin, että kun pelaaja osuu tyhjään ruutuun, hänet palkitaan kerralla paljastuvalla suuremmalla alueella. Varo kuitenkin, `StackOverflowError`saattaa vaania lähistöllä.

Korvaa lopuksi (tai vaikkapa jo hyvinkin varhaisessa vaiheessa osatehtävää) `Pelipaneeli`-luokan `JButton`:it tällaisilla napeilla ja testaa niiden reagointia naksautuksiin.

7. osatehtävä: Game Over

Kun peli päättyy miinan räjähdykseen, paljastetaan kaikkien jäljellä olevien miinojen paikat, eli avataan kaikki vielä avaamattomat miinanapit (ja vain ne – avaamattomat vihjenapit jätetään pimentoon). Se miinanappi, josta räjähdys aiheutui, muuttuu punaiseksi. Näytä lisäksi mahdolliset väärin liputetut ruudut eli miinoiksi merkityt vihjenapit värjäämällä ne vaikkapa violetiksi. Laita myös alapaneelin tekstiksi jokin sopivaksi katsomasi kannustus tai loukkaus.

8. osatehtävä: Pelin voittaminen

Peli on voitettu, kun kaikkien miinojen paikat on selvitetty eli kun kaikki vihjenapit (ja vain ne) on avattu. Lisää pelilogiikasta huolehtivaan `Peliruudukko`-luokkaan metodi `public boolean peliVoitettu()`, joka palauttaa arvon `true`, jos kaikki vihjeruudut (eikä yhtään miinaruutua) on avattu, muuten arvon `false`. Laita käyttöliittymä tarkistamaan voittotilanne jokaisen ruudunavauksen jälkeen. Kun pelaaja voittaa, merkitään lipuilla kaikki loputkin miinat. Kerro pelin voittamisesta myös alapaneelin tekstinä.

9. osatehtävä: Pelin aloittaminen alusta

Liitä ylävalikon Aloita alusta -nappiin toiminto, jolla pelin saa aloitettua uudestaan puhtaalta pöydältä. Voit halutessasi lisätä myös alapaneeliin napin, joka tekee saman asian.

10. osatehtävä: Päheämmät miinat ja liput

Kehitä lopuksi miinoille, lipuille ja virhelipuille hieman hienostuneempi visuaalinen ilme `ImageIcon`-luokan avulla. Voit käyttää Nopasta löytyviä valmiita kuvia tai piirtää itse haluamasi näköiset.

Bonustehtäviä

Seuraavassa on muutamia ehdotuksia, joilla voit vielä parantaa peliäsi mikäli aikaa ja kiinnostusta riittää. Bonustehtäviä ei vaadita hyväksyttyyn suoritukseen, mutta niillä voi korottaa arvosanaansa. Tehtävät ovat suunnilleen helppousjärjestyksessä. Ne ovat toisistaan riippumattomia, eli ne voi tehdä missä järjestyksessä tahansa.

Osa bonustehtävistä vaatii tähänastisen aikaansaannoksemme pientä *refaktorointia* eli ohjelman rakenteen uudelleenjärjestelyä. Tällaisia muutoksia tehdessäsi kannattaa pitää viimeisin toimiva versio ohjelmasta tallessa eri hakemistossa, jotta voit tarvittaessa palata muutoksia edeltäneeseen tilanteeseen.

Fontin vaihtaminen

Swingin käyttämä oletusarvoinen fontti ei välttämättä ole mukavin mahdollinen. Tutki, miten voit muuttaa valikkojen, tekstikenttien ja vihjenumeroiden kirjasintyyppiä `Font`-luokan avulla.

Vaihtelevat vaikeustasot

Laita pelaajalle valittavaksi (esimerkiksi ylävalikosta) muutama erikokoinen ja erisuuruisilla miinalasteilla höystetty pelikenttä. Laita peli-ikkuna myös muuttamaan kokoaan kentän koon mukaan. Pari valmista vaikeusastevaihtoehtoa riittää, koon ja miinamäärän vapaata valintaa ei tarvitse toteuttaa.

Miinalaskuri

Laita sopivaan paikkaan, esimerkiksi alapaneeliin, laskuri, joka näyttää pelin alussa kentässä olevien miinojen määrän ja vähenee sitä mukaa, kun ruutuja merkitään lipuilla – riippumatta siitä, onko liputus oikea vai väärä. Jos lippuja otetaan pois, laskuri vastaavasti kasvaa.

Sovelluksesta sovelmaksi

Kun olet muuten saanut miinaharavapelisi valmiiksi ja olet siihen tyytyväinen, voit vielä kokeilla sen muuttamista www-selaimessa toimivaksi appletiksi eli sovelmaksi. Suosittelemme, että pidät alkuperäisen toimivan sovelluksesi tallessa ja kopioit laatimasi luokat esimerkiksi `applet`-nimiseen alihakemistoon ennen kuin alat muutella niitä. Palauta arvosteltavaksi sovelman ohella myös alkuperäinen `JFrame`-pohjainen sovelluksesi.

Sovelmista löytyy lisätietoa esimerkiksi Sunin How to Make Applets -tutoriaalista.

Lopputoimet

Kun olet ohjelmoinut yllä olevat luokat, testannut niiden toiminnan ja olet muuten tyytyväinen ratkaisuusi, voit palauttaa tehtävän. Palautusohjeet löytyvät Nopasta omasta dokumentistaan.