Javan GUI Scratchaajalle

Asettelu

Tähän mennessä olemme luoneet ikkunan, johon voi laittaa yhden komponentin, kuten nappulan. Yksi komponentti harvoin riittää. Useampaa komponenttia lisäillessä meidän tarvitsee kertoa ikkunalle tarkasti mihin mikäkin komponentti pitää lisätä ja tähän tarvitsemme asettelua.

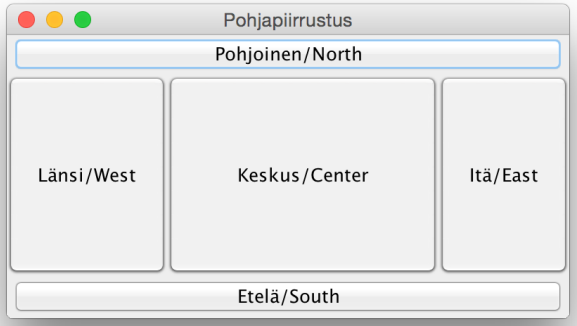
Asettelu eli **layout** on kuin ikkunan pohjapiirrustus.

Java tarjoaa valmiina useita erilaisia asetteluja, jotka osaavat muuntaa ohjelmoijan antaman yksinkertaisen sijaintiselityksen JFrame-ikkunan ymmärtämäksi tiedoksi. JFramelle päätyy tieto siitä, missä komponentin tulee olla ja miten sen kokoon ja paikkaan vaikuttavat muut komponentit.

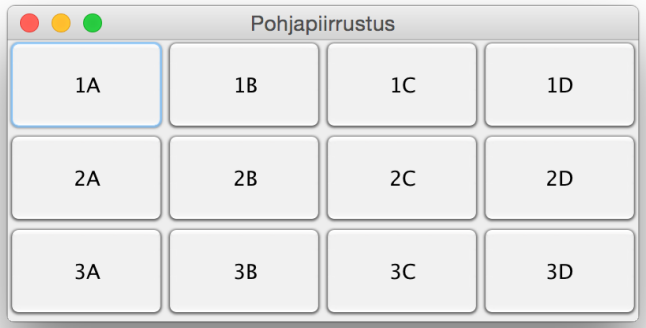
Kuten ikkuna, nappulat ja kuuntelijat, myös asettelut ovat **olioita**.

Erilaisia asetteluja:

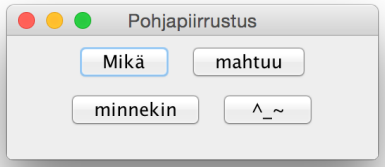
BorderLayout = reunusasettelu



GridLayout = ruudukkoasettelu



FlowLayout = virtaava asettelu



Automaattiasettelu

Jokaisella JFrame -ikkunalla on automaattisesti **BorderLayout** -asettelu, kun se luodaan. Jos siihen lisätään komponentteja ilman “osoitetta”, menevät ne automaattisesti CENTER -alueelle. Tämän takia, jos aiemmin kokeilit lisätä useamman nappulan ikkunaan, menivät ne kukin vuorollaan ikkunan keskelle ja uusi komponentti korvasi aina aiemman lisätyn.

JFrame:n add -metodi osaa ottaa vastaan myös osoitteen:

```
JFrame ikkuna = new JFrame();
ikkuna.add( new JButton(“Länsi/West”), BorderLayout.WEST );
```

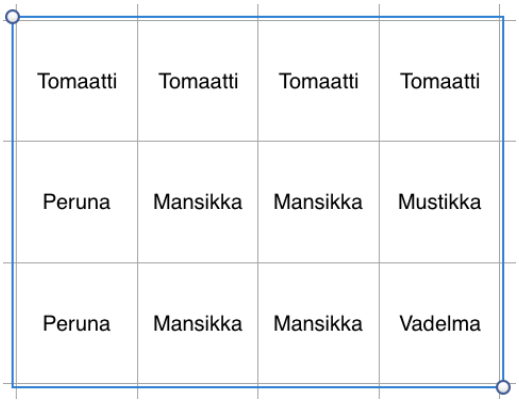
BorderLayout osoitteita ovat:
`NORTH` (pohjoinen), `WEST` (länsi), `CENTER` (keskus), `EAST` (itä) ja `SOUTH` (etelä).

Tehtävä: Puutarha_1

Luo uusi java-projekti. Nimi voi olla vaikka “Puutarhapeli”. Anna Netbeansin luoda Main Class sen nimisenä kuin se ehdottaa.

Puutarhapelissä pelaaja voi kasvattaa ohjelmoijan luomia kasveja. Pelissä on kultaa, jota pelaaja tarvitsee uusien siemenien ostamiseen ja jota pelaaja saa valmiiden kasvisten ym. myynnistä.

Puutarha tullaan istuttamaan ristikkomuodostelmaan:



Tehtävä: Puutarha_2

Tee projektista automaattiajettava eli luo uusi luokka, joka on nimeltään esim. Puutarhanakyma. Tee siitä Runnable ja kirjoita sille toistaiseksi tyhjä run -metodi.

```
public class Puutarhanakyma implements Runnable {
    //tässä välissä voi lukea Netbeansin valmiiksi luomaa tekstiä, älä poista

    @Override
    public void run() {

    }
}
```

Jos teit run-metodin automaattisesti ja tähän tuli `throw new` -rivi, poista se.

Anna **Puutarhapeli**-luokan main-metodissa komento käynnistää graafinen ohjelma:

```
SwingUtilities.invokeLater( new Puutarhanakyma() );
```

Tehtävä: Puutarha_3

Määrittele Puutarhanakyma-luokalle yksityinen (=private) JFrame -muuttuja, jotta mikä tahansa luokan sisäinen metodi, eli komento, voi antaa komentoja JFrame -ikkunalle.

```
public class Puutarhanakyma implements Runnable {
    private JFrame ikkuna;
}
```

Aseta run -metodissa ikkunalle arvoksi `new JFrame();`. Arvon asetus hoituu = -merkin avulla.

Määritä ikkunalle otsikko, koko, sulkumekanismi ja näkyvyys. Testaa, että ikkuna aukeaa, kun projekti ajetaan.

Tehtävä: Puutarha_4

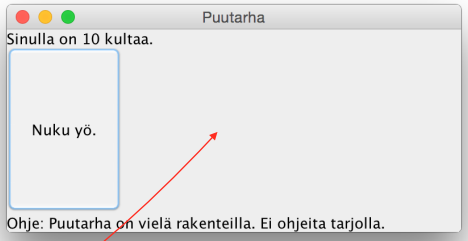
Luodaan ensin pelille ohjeet ja ympäristö ikkunan reunoille. Tee tätä tarkoitusta varten luokkaan uusi metodi Puutarhanakyma-luokkaan:

```
private void rakennaNakyma() {

}
```

1) Kirjoita *rakennaNakyma* -metodiin { } -sulkujen väliin:

- komennot luoda vieressä näkyvän ikkunan kaltaiset nappulat ja etiketit
- komennot ikkunalle lisätä edelliset komponentit näkyviin



Vinkkejä

- Käytä komponentteja lisätessäsi BorderLayout -osoitteita. CENTER-osa on varattu myöhemmin lisättäville istutuksille
- Voit tehdä tekstit etiketteihin tai nappuloihin oman mieltymyksesi mukaan:
  - `new JLabel("etiketti")`
  - `new JButton("nappula")`

2) Kirjoita `rakennaNakyma();` metodikutsu run -metodin loppuun, kun JFrame -ikkuna on jo luotu ja määritelty.

Perintä

Ikkunan keskelle olisi tarkoitus luoda istutuksia. Osa istutuksista voi olla tyhjiä, osassa voi olla kasvava kasvi ja toisissa valmis, myytävissä oleva tuote. Istutuksiin liittyy niin paljon sääntöjä, ettei niitä kannata antaa jokaiselle istutukselle erikseen. Istutuksesta kannattaa siis tehdä luokka, jota voimme sitten kloonata useammaksi olioksi.

Istutuksia pitäisi myös pystyä klikkaamaan, jotta niiden kasveja voisi hoitaa. Tähän soveltuisi JButton -nappula, mutta sillä ei toisaalta ole istutuksien muita ominaisuuksia.

\+ = IstutusNappula

Ongelman ratkaisee perintä. Javassa voimme tehdä istutusnappulan, joka perii JButton -nappulalta kaikki sen ominaisuudet, mutta jolle voimme myös kirjoittaa uusia sääntöjä.

Javassa luokka voi periä toisen luokan extends -komennolla:

```
public class Kissa extends JFrame {

}
```

"Extend" tarkoittaa sanatarkasti laajentamista. Perittyjen ominaisuuksien lisäksi luokalla on nyt perittyyn luokkaan nähden lisää toimintoja, joten sen toimintaa on laajennettu.

Tehtävä: Puutarha_5

Luo Puutarhapeli -projektiin uusi luokka nimeltään Istutus.

Kirjoita Istutus -luokan määritelmään (eli `public class` -riville), että se laajentaa (extends) JButton -luokan toimintaa.

Perivien luokkien alustaminen

Materiaalissa äsken kirjoitettiin, että kissa perii kaikki JFrame -ikkunan ominaisuudet. Tämän jälkeen voisimme aivan hyvin luoda ikkunan aina tekemällä kissan:

```
public void run() {
    Kissa aatuikkuna = new Kissa();
    aatuikkuna.setTitle("Olen kissa nimeltä Aatu.");
}
```

Tämä ei kenties ole aivan tarkoituksen mukaista.

Kun kissa perii JFramen idea on, että uusi luokka on jotenkin erilainen ja parempi kuin peritty luokka. Jo Kissa -luokan luomiskomento voi tehdä ikkunasta kissamaisen. Kissan luomiskomento, eli **konstruktori**, määritettiin Kissa -luokan alussa:

```
public class Kissa extends JFrame {
    //konstruktori on tämä tässä alla
    public Kissa() {
        this.setTitle("Olen kissa nimeltä Aatu.");
        this.setDefaultCloseOperation(JFrame.DO_NOTHING_ON_CLOSE);
    }
}
```

Tehtävä: Puutarha_6

Istutus on tällä hetkellä vain JButton ilman istutusmaisia ominaisuuksia. Kun se luodaan sillä pitäisi olla vähintään istutuksen ulkonäkö.

Luo **pienehköjä*** kuvia: *) 150x150 - 200x200 on kiva
- tyhjän multakeon kuva
- kuvia yhden kasvin eri vaiheista (multakeko on taustana)

Huom!
Tee kuvista **samankokoiset**. Tee alkuperäisestä multakeosta siis kopioita ja piirrä niiden päälle eri vaiheet. Tällöin kasvi ei kasvaessaan näytä pomppivan ruudulla.

Tallenna kuvat projektin kansioon (esim. NetBeansProjects > Puutarhapeli).

Emme saa kuvia ihan vielä näkyviin, mutta luo `Istutus` -luokalle private kasvuvaihe-muuttuja ja konstruktori, jossa vaihe asetetaan nollaan:

```
private int kasvuvaihe;

public Istutus() {
    this.kasvuvaihe = 0;
}
```

Listat

Pelissä luomme seuraavaksi useita kuvia nappulalle. Kuvien käsitteleminen yksitellen olisi hankalaa, joten luodaan niistä lista. Scratchissa listat tehtiin kuten muuttujatkin, mutta Javassa ne ovat olioita. Helpoin lista käyttää on ArrayList. Sen pituus kasvaa automaattisesti kun uusia lista-alkioita lisätään.

ArrayList tehdään pitkälti kuten muutkin oliot, mutta sille määritellään myös minkä tyyppisiä alkioita siihen talletetaan. Vertaa:

Ikkunan luonti:
```
JFrame ikkuna = new JFrame();
```

Listan luonti:
```
ArrayList<String> lista = new ArrayList<String>();
```

Jos haluaisimme tehdä listan numeroita, pitäisi `String` tekstin tilalle kirjoittaa `Integer` (int ei riitä). Toisaalta listaan voisi tallettaa myös Kissa-, JButton- ja JFrame-olioita. Tai miksei myös listoja.

> **Tehtävä: Puutarha_7**
>
> Tee Istutus -luokalle private muuttuja, joka on ArrayList. Tämän listan sisään laitetaan kohta ImageIcon -olioita, joten määrittele se < > sulkujen väliin. Listan nimi voi olla vaikkapa `kuvat`.
>
> Huom! Tällaiset luokkakohtaiset muuttujat määritellään siististi ohjelmoitaessa luokan alussa yhdessä paikassa. Laita tämä rivi siis jo kirjoitetun `private int kasvuvaihe` -rivin alle.

Lista-alkioiden lisäys

Tyhjä lista ei meitä auta. Onneksi listalle voi antaa komennon ottaa vastaan sille sopivia oliota. Tämä komento on `add` (eli lisää).

```
ArrayList<JButton> nappulat = new ArrayList<JButton>();
nappulat.add(new JButton("Pää"));
nappulat.add(new JButton("Tassut"));
nappulat.add(new JButton("Häntä"));
```

Lista-alkioiden tiedustelu

Aiemmin, jos meillä oli nappula, pystyimme liittämään sen komentoihin, kun tiesimme sen nimen:

```
JButton kaninappula = new JButton("Olen kani");
System.out.println( kaninappula );
```

Nyt nappulat ovat listassa, joten meidän tulee komentaa listaa antamaan jossain tietyssä kohdassa oleva alkio. Voimme liittää sen suoraan komentoihin:

```
ArrayList<JButton> kerhot = new ArrayList<JButton>();
kerhot.add(new JButton("Scratch-kerho"));
kerhot.add(new JButton("Java-kerho"));
kerhot.add(new JButton("Algoritmikerho"));

System.out.println( kerhot.get(0) );

kerhot.get(0).addActionListener( new NappulaKuuntelija() );
```

Bonus: Huomasit kenties, ettei nappuloita ensin tehty omalla `JButton kerhonappula = new JButton("Java-kerho");` -rivillä. Tämä on välttämätöntä silloin kun haluamme talteen myöhempiä komentoja varten muuttujan nimen, eli nyt `kerhonappula`. Aina nimeä ei tarvitse muistaa, jos esim. listan nimen kautta oikean nappulan saa helposti komennettavaksi. Tällöin voi yksinkertaisesti luoda uuden olion: `new JButton("Java-kerho");`

Tehtävä: Puutarha_8

Luo luokan konstruktorissa (construct = rakentaa), eli public Istutus() -rivin jälkeisten { } -sulkujen sisässä, ImageIcon jokaiselle tekemällesi kuvalle. Lisää tehdyt ImageIcon -oliot edellisessä tehtävässä tekemääsi listaan kasvujärjestyksessä.

ImageIcon luotiin komennolla: `ImageIcon kuva = new ImageIcon("Nimi.png");`

Halutessasi voit lisätä ImageIcon -oliot jo suoraan tehtäessä listaan.

Huom. Kuvat piti tallentaa NetBeansProjects -kansioon tämän projektin kansioon, jotta peli löytää ne. Huomaa myös, että Scratchia käyttäessäsi tallenna asuste, älä hahmoa.

Tehtävä: Puutarha_9

Aseta konstruktorissa listan luomisen jälkeen nappulan kuvaksi listan ensimmäinen ImageIcon.

Kuva asetettiin komennolla `setIcon(kuva)`. Kuvan saat käytettäväksi `get` -komennolla.

Komennon eteen ei ole välttämätöntä tällä kertaa laittaa komennettavaa hahmoa (kuten `nappula.setText("Uusi teksti");`), koska annamme komennon suoraan luokalle, johon kirjoitamme komennon. Jos komennettavan hahmon haluaa mainita niin se on `this`, eli kirjaimellisesti "tämä".

Tehtävä: Puutarha_10

Nyt kun istutuksella pitäisi olla kuva, sitä olisi myös kiva kokeilla. Laitetaan aluksi pellolle vain yksi istutus.

3. Tee Puutarhanakyma -luokkaan myös uusi `private void kynnaKasvimaa()` -metodi ja kutsu sitä, eli kirjoita komennon nimi ja sulut perään, `run` -metodissa `rakennaNakyma()` -komentokutsun perään.

4. Tee `kynnaKasvimaa` -metodin sisällä uusi (=new) Istutus -olio ja lisää se luotuun ikkunaan keskelle.

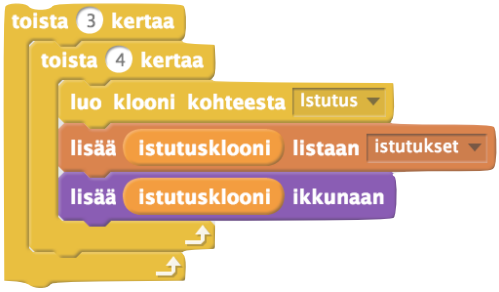
Testaa, että kuva näkyy.

Tehtävä: Puutarha_11

Korjataan nyt kynnaKasvimaa -metodia niin, että siinä luodaan yhden istutuksen sijaan kaikki halutut. Koska istutuksia tulee kuitenkin paljon, on kannattavaa käyttää jälleen listoja!

1. Tee Puutarhanakyma -luokkaan luokkakohtainen `private` muuttuja, joka on ArrayList. Tähän listaan laitetaan `Istutus` -olioita, joten määritä se < > -sulkujen väliin. Muuttujan nimi voi olla vaikka `istutukset`.

2. Tee kynnaKasvimaa -metodiin viereisen kaltainen ohjelma. Listaan lisäyksen jälkeen lisää istutus myös ikkunaan keskelle.



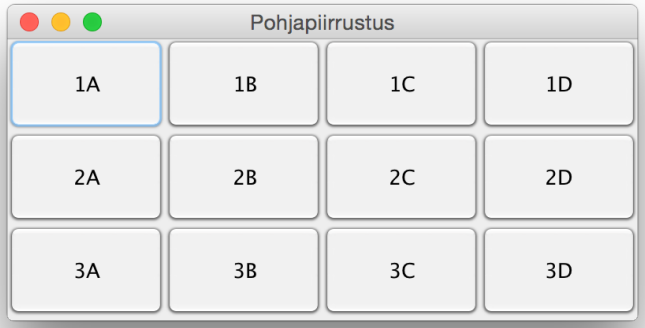
Kasvimaa ei näytä vieläkään enempää kuin yhden istutuksen. Korjataan se seuraavaksi.

Asettelun määrittely

Luotujen ikkunoiden automaattinen asettelu on BorderLayout. Ikkunoille voi kuitenkin antaa myös komennon asetella komponenttinsa erilailla. Tähän käytetään `setLayout` -komentoa:

```
JFrame ikkuna = new JFrame();
//ohjelmoi tähän väliin ikkunan määritys

int rivit = 3;
int sarakkeet = 4;
ikkuna.setLayout( new GridLayout(rivit,sarakkeet) );
```



Asetteluksi voi laittaa myös muita:

```
ikkuna.setLayout( new BorderLayout() );
ikkuna.setLayout( new FlowLayout() );
ikkuna.setLayout( new BoxLayout(ikkuna, BoxLayout.X_AXIS) );
```

Kukin asettelu voi tarvita, tai olla tarvitsematta, taustatietoa. GridLayout tarvitsi rivien ja sarakkeiden määrän, BoxLayout ikkunan, jossa se on ja laatikoiden asettelusuunnan.

Ikkunan eri alueiden asettelu

Ikkuna ei ole ainoa alue, jolla voi olla asettelu. Ikkunaan voi lisätä myös “ikkunapaneeleita”, joilla voi olla oma asettelu.

```
JFrame ikkuna = new JFrame();
//ohjelmoi tähän väliin ikkunan määritys

JPanel paneeli = new JPanel();
paneeli.setLayout( new GridLayout(1,4) );
```

Koska paneeli on käytännössä uusi, pienempi ikkuna, voi se jälleen sisältää lisää komponentteja. Ne lisätään `add` -komennolla.

Ajatusharjoitus:

Voit käytännössä ajatella, että JFrame ja JPanel ovat laatikoita siinä missä JButton, JLabel ja JTextField ovat tavaroita. Tavaroita voi laittaa laatikoiden sisälle.

Toisaalta myös laatikot ovat tavaroita, joten laatikoita voi laittaa laatikoiden sisälle.

Kullekin laatikolle voi Java-maailmassa määritellä oman asettelunsa.

> Tehtävä: Puutarha_12
>
> Luo **`kynnaKasvimaa`** -metodissa ennen for-silmukoita uusi JPanel ja aseta (set) sille asetteluksi uusi GridLayout. Voit asettaa rivejä esim. 3 kpl ja sarakkeita 4 kpl. Paneelin nimi voi olla vaikka `kasvimaa`.
>
> For-silmukoiden sisällä korjaa ohjelmaa niin, että istutukset lisätään kasvimaalle, ei ikkunaan.
>
> **`kynnaKasvimaa`** -metodin lopussa lisää kasvimaa-JPanel ikkunaan sen keskelle (CENTER).

Ikkunan ulkonäöstä

Nappulat

JButton näyttää tyypillisesti nappulalta. Sille voi kuitenkin antaa sen ulkonäköön liittyviä komentoja.

`setOpaque(true)`	aseta läpinäkymättömäksi (totta)
`setBorderPainted(false)`	aseta reuna piirretyksi (epätotta)
`setContentAreaFilled(false)`	aseta sisusalue väritetyksi (epätotta)
`setFocusPainted(true)`	aseta fokus piirretyksi (totta)

Värit

Javassa värit osaa parhaiten Color. Sen voi “lainata” projektiinsa, kun luokan alkuun kirjoittaa

```
import java.awt.Color;
```

Siltä voi pyytää valmisvärejä kuten Color.WHITE ja Color.BLUE. (Löydät lisää googlettamalla “color java api”.) Uusia värejä voi myös tehdä kun antaa vain haluamansa värin punaisen, vihreän ja sinisen määrät:

```
Color vari = new Color(30, 190, 80);
```

> Tehtävä: Puutarha_13
>
> Määritä ikkunasi koko uudelleen, jotta kasvimaa sopii sille.
>
> Määritä myös istutusten ulkonäköä ja kasvimaan taustaa. Kasvimaalle voit antaa taustavärin komennolla kasvimaa.setBackground(Color.GREEN) tai Color.GREEN kohtaan voi myös kirjoittaa uuden värin: new Color(…)
>
> Jotta kasvimaan taustaväri näkyisi nappuloiden alta, karsitaan nappuloiden piirrosta pari osaa. Karsimiskomennot voisi antaa kullekin nappulalle erikseen kynnaKasvimaa -metodissa, mutta koska tämä on enemmän istutuksen perusominaisuus, on parempi kirjoittaa komennot suoraan sen konstruktoriin. Anna siis Istutus-luokan konstruktorissa sille (this) aiempia läpinäkyvyyteen, reunoihin ja sisukseen liittyviä komentoja.
>
> Minkä ominaisuuksien haluat olevan päällä (true) ja minkä poissa päältä (false)?