Javan alkeet Scratch-ohjelmoijille

1. Tekstiseikkailun aloittaminen

Osaamme nyt sen verran Javan alkeita, että voisimme tehdä jonkin pelin. Ensimmäinen pelimme on tekstiseikkailu.

Tehtävä: Tekstiseikkailu_1

Luo uusi projekti. Projektia luodessa valitse taas ensin “Next” ja kirjoita sitten projektin nimen “JavaApplication1” tilalle esimerkiksi “Tekstiseikkailu”. Paina “Finish”.

Tulemme tekemään kaikki tämän monisteen tehtävät tähän projektiin. Materiaalissa on kuitenkin useita esimerkkejä. Esimerkkejä varten kannattaa tehdä omia miniprojekteja.

Tehtävä: Tekstiseikkailu_2

Poista `// TODO code application logic here` -rivi. Luo kyseisellä rivillä ohjelmaasi lukija kuten materiaalin ensimmäisessä osassa opastettiin. Lainaa ohjelman käyttöön tarvittu Scanner -olio, jotta punaisella alleviivattuja sanoja ei enää ole.

Tehtävä: Tekstiseikkailu_3

Aloitetaan tekstiseikkailu kysymällä tietoja pelaajasta. Scanneria voi käyttää vasta sen jälkeen kun se on luotu, joten kirjoita seuraavat rivit ***`main`*** -tapahtuman sisään Scannerin luomisen perään:

```
System.out.println("Kerro nimesi.");
String nimi = lukija.nextLine();
System.out.println("Anna ikäsi.");
int ika = Integer.parseInt(lukija.nextLine());
System.out.println("Tervehdys sinulle " + nimi + ". " + ika + " on hyvä olla.");
```

Katso miten käy. Muista, että sanoja voi tulostaessa tosiaan liimata toisiinsa + -merkein. `nimi` ja `ika` tulevat ilman hipsuja, koska niille on ohjelmassa annettu `=` -merkillä erityismerkitys.

Huomaa myös, että välilyönnitkin kannattaa kirjoittaa hipsujen väliin, jos ne haluaa tulostukseen.

Alkuperäinen tekijä: Ada-Maaria Hyvärinen (2016), muokannut: Jenna Tuominen (2016), muokannut: Milla Kortelainen ja Jenna Tuominen (2019)

Haluamme todella usein koodatessa tulostaa eli käyttää komentoa `System.out.println("");`. Onneksi käytössämme on pikakomento. Kirjoita riville ensin merkkijono `sout` ja paina sarkainpainiketta (⇥). Tämä komento täytyy kirjoittaa kerralla oikein, muuten se ei toimi, eli jos teet kirjoitusvirheen kesken komennon kirjoittamista, sinun täytyy aloittaa alusta.

Tehtävä: Tekstiseikkailu_4

Tarkista, että ohjelmasi sisennykset ovat oikein.

2. Matikkaa

Javalla voi tehdä laskuja tavalliseen tapaan merkeillä **+**, **-**, ***** ja **/**. Merkillä **%** saa tietää jakojäännöksen.

Tehtävä: Tekstiseikkailu_5

Kirjoita ***main*** -tapahtuman alle, edellisten komentojen perään seuraavat komennot:

```
System.out.println("Kello käy jo " + 2 + 2 + ",");
System.out.println("ja polkuja haarautuu tästä " + (1 + 1) + ".");
```

Miten laskutoimistusten tulokset eroavat? Kummassa tapauksessa `+` -merkki vastaa Scratchin + -merkkiä, ja kummassa Scratchin yhdistä -palikkaa?

Laskujärjestys

Edellisessä tehtävässä laskut tulostuivat aivan erilailla. Toisessa luvut laskettiin oikeasti yhteen ja toisessa ne vain tulostuivat peräkkäin. Tämä johtuu Javan laskujärjestyksestä. Koska laskujärjestyksessä ensin lasketaan yhteen `"tekstiä" + luku`, valitsee Java vastaukseksi tekstin. Jos haluaa, että Java laskee luvut yhteen ensin, pitää niiden ympärille laittaa sulut.

Toki, jos laskussa lasketaan yhteen vain lukuja eikä mukana ole tekstiä, ei ylimääräisiä sulkuja tarvitse.

3. Ehtolauseet

Ehtolauseet ovat ohjelmoinnissa työkaluja, joilla saamme ohjelman toimimaan erilailla eritilanteissa. Javassa ehtolauseeseen kuuluu `if`, jonka perässä on suluissa jokin väittämä, joka voi olla totta tai tarua. Jos ehto oli totta, suoritetaan {} -sulkujen väliin asetetut komennot.

```
int valinta = 1;
if (valinta == 1) {
        System.out.println("Nyt kävi huonosti");
}
```

Tehtävä: Tekstiseikkailu_6

Kirjoita ohjelmasi loppuosaan seuraava koodi:

```
System.out.println("Käytkö polulle 1 vai 2?");
int valinta = Integer.parseInt( lukija.nextLine() );

if (valinta == 1) {
    System.out.println("Sankarin niin urhean,\nsöi kuitenkin lopulta hirviö.");
}
else {
    System.out.println("Ja näin kannatti koittaa,\noli linnan ovi auki.");
}
```

Selvitä mitä tämä pätkä ohjelmaa tekee. Kirjoita vastaus kommentiksi koodiisi sopivaan kohtaan.

Javassa (ja Scratchissa) voi `if`:n perään lisätä myös "`else`" eli "muuten" -lohkon. Java tulkitsee tämän niin, että jos `if`:n sisällä oleva ehto ei täyty niin suoritetaankin `else`-lohkon sisään kirjoitetut komennot.

```
int kissoja = 2;
if (kissoja == 1) {
        System.out.println("Kissoja oli yksi.");
}
else {
        System.out.println("Kissoja ei ollut yksi vaan joku eri määrä.");
}
```

Huomaa, että `else`-lohkon yhteydessä ei kirjoiteta mitään ehtoa ()-sulkujen sisään. Tämä johtuu siitä, että `else`:en mennään joka tapauksessa jos `if` ei täyttynyt. `else`:en ei liity mitään erityisiä lisäehtoja.

Alkuperäinen tekijä: Ada-Maaria Hyvärinen (2016), muokannut: Jenna Tuominen (2016), muokannut: Milla Kortelainen ja Jenna Tuominen (2019)

Javassa voi tehdä myös “muussa tapauksessa jos” -ehtoja `if`-lohkon jälkeen:

```
Scanner lukija = new Scanner(System.in);
int valinta = Integer.parseInt( lukija.nextLine() );
if (valinta == 1) {
	System.out.println(“Hirviö tuli ja söi sinut”);
}
else if (valinta == 2 ) {
	System.out.println(“Tämä olikin umpikuja”);
}
else if (valinta == 3 ) {
	System.out.println(“Löysit prinsessan!”);
}
else {
	System.out.println(“Eksyit metsään”);
}
```

`else if` on kohta, johon ohjelman suoritus menee, jos edellinen ehto ei ollut totta. Edellisessä esimerkissä, jos valinta ei ollut 1 testaa ohjelma seuraavaksi onko se 2. Jos tämäkään ei ole totta, siirrytään seuraavaan `else if` -kohtaan. Jos ohjelmaan halutaan jokin `else` -tapaus, pitää se aina kirjoittaa viimeiseksi. Huomaa, että `else if` -lohkoja voi olla useampia, mutta `else` -lohkoja vain yksi!

Lukujen vertailu

Javassa vertailemme lukujen samanarvoisuutta tai suuruus eroa käyttämällä matematiikasta tuttuja merkkejä.

Merkki	Selitys
==	Yhtäsuuri kuin
<	Pienempi kuin
<=	Pienempi tai yhtäsuuri kuin
>	Suurempi kuin
>=	Suurempi tai yhtäsuuri kuin

Huomaa, että vertailemme kahden luvun yhtäsuuruutta käyttämällä kahta yhtäsuuruusmerkkiä. Yhdellä yhtäsuuruusmerkillä Java asettaa muuttujalle uuden arvon. Kahta merkkijonoa ei voi verrata ihan yhtä helposti, sitä harjoittelemme vähän myöhemmin.

Tehtävä: Tekstiseikkailu_7

Korjaa aiempaa koodia niin, että polkuja on kolme. Saat tehtyä tämän äsken esitellyllä `else if` -lohkolla. Keksi mitä tällä polulla käy.

Sisäkkäiset ehdot

`if`-ehdon sisälle voi laittaa myös toisen `if`-ehdon. Tällöin sisällä oleva `if`-ehto toteutuu vain, kun ulkopuolella oleva `if`-ehto on toteutunut ensin. Sisäkkäisen `if`-ehdon voi laittaa myös `else if` tai `else` -lohkon sisään. Tällöin nämä sisemmät ehdot toteutuvat, kun ulkopuolella oleva `else if` tai `else` toteutuu.

```
else {
        System.out.println("Eksyit metsään");
        if (ika < 10) {
                System.out.println("Onneksi tunsit kuitenkin metsän.");
        }
}
```

Tämän sisemmän `if`:n perään voi sitten halutessaan laittaa myös siihen liittyviä `else if` ja `else` -lohkoja.

Tehtävä: Tekstiseikkailu_8

Ohjelmassasi pitäisi nyt olla yksi `if`, `else if` ja `else`. Kirjoita seuraava pätkä ohjelmaa tuon `else`:n sisään, jotta tarinan sankari pääsee jatkamaan matkaansa.

```
System.out.println("Näkyi ovelta portaat ja sali,\nkumman valitsee sankari?");
String kohde = lukija.nextLine();
if ( kohde.equals("sali") ) {
        System.out.println("Kuului kauhea karjaisu,\npelästytti pois sankarin.");
}
else {
        System.out.println("Askelmat jyrkät,\nselvisi täpärästi." + nimi);
}
```

Muista, että kun muuttuja on kerran luotu, ei sitä voi luoda uudelleen. Et siis vastausta lukiessasi voi aina tehdä samannimistä muuttujaa uudelleen. Muuttujaan voi kuitenkin myöhemmin tallettaa jonkin eri arvon:

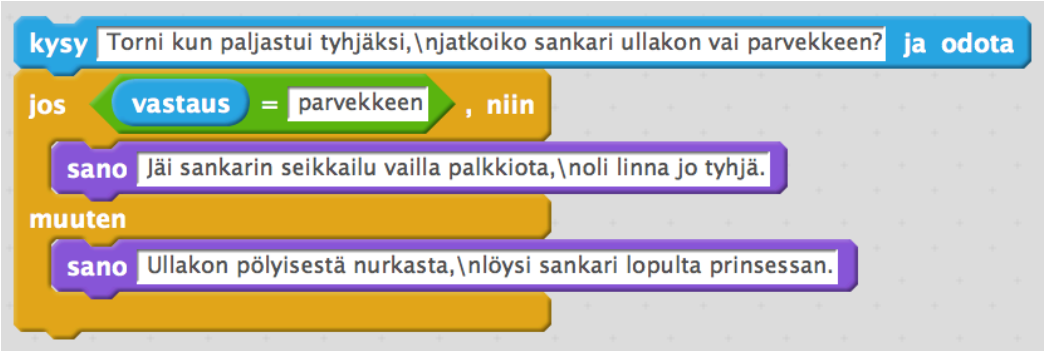
```
int palloja = 2;
System.out.println("Tuutti " + palloja + " pallolla, ole hyvä.");

palloja = 5;
```

Jos haluat siis esimerkiksi käyttää `kohde` -muuttujaa pelaajan vastauksen lukemiseen myöhemmin uudelleen, älä kerro sille uudelleen, että se on `String`.

Tehtävä: Tekstiseikkailu_9

Lisää vielä linnan torniin haaste ja palkinto. Laita tämä johonkin sopivaan `if`, `else if` tai `else` -lohkoon ohjelmassasi sen mukaan mikä reitti on oikea. Esimerkissämme alla oleva “torni” on edellisessä tehtävässä lisättyjen portaiden päässä.

Muunna Scratch-koodi Javaksi. Katso tarvittaessa apua edellisistä tehtävistä. Kysy -palikka pitää Javassa tehdä kahdella rivillä. Kysymys pitää ensin tulostaa käyttäjälle ja vastaus voidaan lukea Scannerilla vasta sen jälkeen.



Valinnainen: Tekstiseikkailu_10

Jatka tarinaa niin monimutkaiseksi kuin haluat.

Bonus: Virheilmoitukset

Scratch on hyvin sallivainen ohjelmointikieli, ohjelma toimii jopa vaikka sitä käyttäisi väärin. Javassa virheitä voi taas syntyä niin monesta erilaisesta asiasta, että niitä voi ryhmitellä eri otsikoiden alle. Tutustutaan nyt ajonaikaisiin virheisiin.

Ajonaikaisia virheitä voi syntyä vaikka koodissa ei olisi mitään vikaa. Tällainen virhe voi tulla esimerkiksi jostain käyttäjän kirjoittamasta vastauksesta. Näihin virheisiin voi varautua ja yleensä kannattaakin ellei halua ohjelman kaatuvan.

Alkuperäinen tekijä: Ada-Maaria Hyvärinen (2016), muokannut: Jenna Tuominen (2016), muokannut: Milla Kortelainen ja Jenna Tuominen (2019)

Virheviesti voi näyttää esim. tältä:

```
Exception in thread "main" java.lang.NumberFormatException: For input string: "virhe"
        at java.lang.NumberFormatException.forInputString(NumberFormatException.java:65)
        at java.lang.Integer.parseInt(Integer.java:580)
        at java.lang.Integer.parseInt(Integer.java:615)
        at tekstiseikkailu.Tekstiseikkailu.main(Tekstiseikkailu.java:26)
Java Result: 1
BUILD SUCCESSFUL (total time: 11 seconds)
```

Virheviestejä luetaan erityisellä tavalla. Virheviestin ensimmäisellä rivillä kerrotaan mikä virhe tapahtui. Esimerkin tapauksessa syntyi `NumberFormatException`. Käyttäjä on ilmeisesti kirjoittanut vastaukseksi “virhe” kysymykseen, josta ohjelma oletti vastaukseksi lukua.

Ensimmäisen rivin jälkeisillä riveillä Java osoittaa mitä reittiä pitkin virhe syntyi niin, että rivi johon ohjelma sitten kaatui on ylimpänä. Tästä polusta voi usein jättää huomiotta vieraannäköiset rivit. Tässä polussa ensimmäinen tutunnäköinen rivi on “at tekstiseikkailu.Tekstiseikkailu.main”, se on hyödyllinen. Sillä rivillä kerrotaan, että virhe syntyi Tekstiseikkailu-tiedostossa rivillä 26. Näin ohjelmoija voi etsiä mikä osa ohjelmaa kaipaa kenties vielä viilausta.

Bonustehtävä: Virheet_1

Käytä tähän tehtävään esim. tekstiseikkailuun jo kirjoittamaasi koodia. Mikä virhe syntyy, jos vastaat ohjelmalle, että ikäsi on jotain mikä ei ole numero?

Miksi tämä virhe syntyy?

Bonustehtävä: Virheet_2

Mikä virhe syntyy, kun ajat ohjelman, josta löytyy laskutoimitus: `(2 / 0)` ?

Voit käyttää tähänkin tehtävään esim. tekstiseikkailuun jo kirjoittamaasi koodia, jos muokkaat siinä seuraavaa riviä:

```
System.out.println("ja polkuja haarautuu tästä " + (1 + 1) + ".");
```

Miksi tämä virhe syntyy?