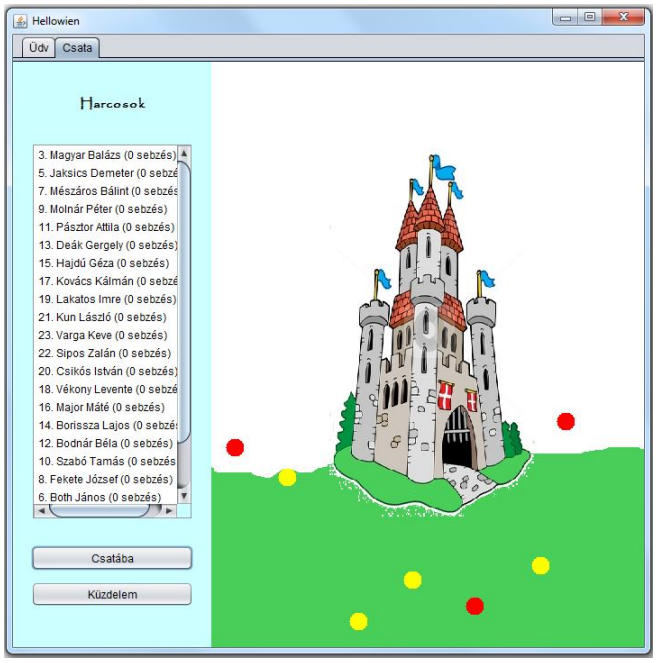
Hello Wien – szálkezelő része



Harcoljanak Mátyás katonái!
Vagyis a csatába küldés után változzon aktívra a „Küzdelem” feliratú gomb (előtte nyilván inaktív volt), hatására pedig bizonyos időközönként menjen arrébb egy-egy, a harcolók közül véletlenül kiválasztott katona. Az „arrébb menni” most azt jelenti, hogy egy másik véletlen helyre ugrik.

A küzdelmet csak egyszer lehet elindítani, ezért megnyomása után a gomb felirata változzon meg „Leállít”-ra, és ha ismét megnyomjuk, álljon le a csata.
(Segítség: Célszerű a vezérlő osztályt átalakítani szál osztállyá.)

FONTOS: Ha a vezérlő osztály működését korábban a `start()` metódus indította, akkor most kezdje avval, hogy ezt átnevezi modjuk `indit()`-ra (Refaktorizációval.)

Egy lehetséges megoldásrészlet

A `HarcosokPanel` osztályban egyetlen bővítés van, a Küzdelem gombhoz tartozó esemény:

```
private void btnKuzdelemActionPerformed(java.awt.event.ActionEvent evt) {
    if (btnKuzdelem.getText().equals(EREDETI)) {
        vezerlo.kuzdelem();
        btnKuzdelem.setText(UJ);
    }
    else{
        vezerlo.leall();
    }
}
```

Az összes többi a `Vezerlo` osztály dolga.

Mivel a vezérlő most időnként arrébb rak egy-egy kiválasztott harcost, ezért az osztályt szálként definiáljuk, azaz a `Thread` osztály leszármazottjaként. Emiatt megismétlem a feladatkiírásban már szereplő nagyon fontos megjegyzést: ha a `Vezerlo` osztályban esetleg van `start()` nevű metódus, akkor azt egy refaktorizációval feltétlenül írja át valami másra, mondjuk legyen `indit()`. Ez azért fontos, mert különben felüldefiniálnák a `Thread` osztály `start()` metódusát, és nem lenne, aki elindítsa a szálat.

A `Vezerlo` osztály `kuzdelem()` metódusának az a szerepe, hogy elindítsa a szálat, azaz jelen pillanatban meghívja a saját `start()` metódusát.

```
public void kuzdelem() {
    if(!this.isAlive()){
        kuzdelem = true;
        this.start();
    }
}
```

A `run()` metódusban kell megírnunk, hogy mi is történjen a szál hatására. Összesen annyi, hogy amíg tart a küzdelem, addig bizonyos időközönként a „király” válasszon ki egy véletlen harcost és állítsa egy véletlen helyre. Természetesen csak akkor, ha egyáltalán vannak harcosok.
Vagyis ugyanazt ismétli meg egy kiválasztott harcossal, mint a csata kezdetén az összes harcossal. Ezt korábban a `csatabaAllit()` metódusban oldottuk meg. A kódismétlés elkerülése céljából a megismétlendő részletet kirakjuk egy külön metódusba. Így a következő-képpen alakul a metódus:

```
/**
 * Csatába küldi a kiválasztott katonákat, azaz mindegyiknek megmondja,
 * hogy hova álljon - ez a csatatér egy véletlen pontja.
 *
 * @param valasztottKatonak
 */
public void csatabaAllit(List<Katona> valasztottKatonak) {
    this.harcosok = valasztottKatonak;
    for (Katona katona : harcosok) {
        beall(katona);
    }
    // frissítjük a csatapanelt
    csataPanel.repaint();
}

private void beall(Katona katona) {
    katona.setKx((int) (Math.random() * (csataPanel.getWidth()
            - 2 * Katona.getSugar()) + Katona.getSugar()));
    katona.setKy((int) (Math.random() * (csataPanel.getHeight()
            - rajzMagassag - Katona.getSugar()) + rajzMagassag));
}
```

Ezek után a `run()` metódus:

```
@Override
public void run() {
    int index;
    while(kuzdelem){
        if(!harcosok.isEmpty()){
            try {
                index = (int) (Math.random()*harcosok.size());
                beall(harcosok.get(index));
                csataPanel.repaint();
                Thread.sleep(ido);
            } catch (InterruptedException ex) {
                Logger.getLogger(Vezerlo.class.getName()).log(Level.SEVERE, null, ex);
            }
        }
    }
}
```

A leállítás:

```
    public void leall() {
        kuzdelem = false;
    }
```