MAC0321 – Trabalho em Grupo – Star Wars

Alair Pereira do Lago
Alexandre Silveira Moreira de Toledo
Thâmillys Marques de Oliveira

17 de maio de 2018

Instruções

Esse trabalho pode ser feito em duplas e deve ser feito durante o horário da aula. Vocês terão duas aulas para resolver os dois exercícios descritos, porém caso a dupla ache necessário é possível fazer parte do exercício em casa também.

A entrega **não** será feita pelo PACA. Os alunos terão que criar uma conta no GitHub[1] ou BitBucket[2] (pode ser apenas uma conta por dupla, mas preferencialmente cada membro da dupla cria uma conta e os dois têm acesso de escrita no repositório) e colocam todo o código nesse repositório. Esse repositório deve ser público, pois no final iremos baixar o código dos repositórios para correção. **Recomendação:** não deixem para *commitar* apenas quando tudo estiver pronto! Façam *commits* constantes, isso evita que vocês percam parte do trabalho. No Git, além de commitar vocês DEVEM dar um **push** nos commits para que as atualizações vão para o repositório remoto.

Introdução

Observe a forma utilizada por Bruce Eckel[3] para escrever um esqueleto de programa que controla uma *Greenhouse* (estufa para plantas).

Primeiro foi criada uma classe abstrata que serve de base para os eventos, esta base é responsável pela definição de quais métodos devem estar disponíveis para os eventos:

```
1 package c07.controller;
2
3 abstract public class Event {
```

[1] https://github.com/
[2] https://bitbucket.org/
[3] Thinking in Java, disponível na internet no endereço: www.bruceeckel.com

```
4   private long evtTime;
5   public Event(long eventTime) {
6     evtTime = eventTime;
7   }
8   public boolean ready() {
9     return System.currentTimeMillis() >= evtTime;
10  }
11  abstract public void action();
12  abstract public String description();
13} ///:~
```

Vemos claramente que nesta classe existem 4 métodos, sendo dois deles abstratos (`action` e `description`). No método `Event` (construtor), o instante a partir do qual o evento pode ser disparado é determinado. O método `ready` apenas retorna verdadeiro caso o objeto evento em questão está "pronto".

A partir desta classe `Event`, uma classe para controlar vários objetos do tipo Event é criada:

```
1 package c07.controller;
2
3 // This is just a way to hold Event objects.
4 class EventSet {
5   private Event[] events = new Event[100];
6   private int index = 0;
7   private int next = 0;
8   public void add(Event e) {
9     if(index >= events.length)
10      return; // (In real life, throw exception)
11    events[index++] = e;
12  }
13  public Event getNext() {
14    boolean looped = false;
15    int start = next;
16    do {
17      next = (next + 1) % events.length;
18      // See if it has looped to the beginning:
19      if(start == next) looped = true;
20      // If it loops past start, the list
21      // is empty:
22      if((next == (start + 1) % events.length)
23         && looped)
24        return null;
```

```
25      } while(events[next] == null);
26      return events[next];
27    }
28    public void removeCurrent() {
29      events[next] = null;
30    }
31  }
32
33  public class Controller {
34    private EventSet es = new EventSet();
35    public void addEvent(Event c) { es.add(c); }
36    public void run() {
37      Event e;
38      while((e = es.getNext()) != null) {
39        if(e.ready()) {
40          e.action();
41          System.out.println(e.description());
42          es.removeCurrent();
43        }
44      }
45    }
46  } ///:~
```

Na classe `EventSet` vemos que existem três membros privados: `index` que corresponde à quantidade de eventos no vetor de eventos `Event`, e `next` que corresponde ao evento em questão. Existem também métodos para adicionar um evento ao vetor, pegar o próximo evento (não nulo) do vetor e remover o evento em questão.

A classe `Controler` gerencia um elemento da classe `EventSet`. Seus métodos permitem a adição de um elemento, e a varredura dos eventos, enquanto existirem eventos disponíveis.

Finalmente na classe `GreenhouseControls` serão criados eventos para controlar uma estufa.

```
1   package c07.controller;
2
3   public class GreenhouseControls extends Controller {
4     private boolean light = false;
5     private boolean water = false;
6     private String thermostat = "Day";
7     private class LightOn extends Event {
8       public LightOn(long eventTime) {
9         super(eventTime);
```

```
10      }
11      public void action() {
12        // Put hardware control code here to
13        // physically turn on the light.
14        light = true;
15      }
16      public String description() {
17        return "Light is on";
18      }
19    }
20    private class LightOff extends Event {
21      public LightOff(long eventTime) {
22        super(eventTime);
23      }
24      public void action() {
25        // Put hardware control code here to
26        // physically turn off the light.
27        light = false;
28      }
29      public String description() {
30        return "Light is off";
31      }
32    }
33    private class WaterOn extends Event {
34      public WaterOn(long eventTime) {
35        super(eventTime);
36      }
37      public void action() {
38        // Put hardware control code here
39        water = true;
40      }
41      public String description() {
42        return "Greenhouse water is on";
43      }
44    }
45    private class WaterOff extends Event {
46      public WaterOff(long eventTime) {
47        super(eventTime);
48      }
49      public void action() {
50        // Put hardware control code here
51        water = false;
```

```
52      }
53      public String description() {
54        return "Greenhouse water is off";
55      }
56    }
57    private class ThermostatNight extends Event {
58      public ThermostatNight(long eventTime) {
59        super(eventTime);
60      }
61      public void action() {
62        // Put hardware control code here
63        thermostat = "Night";
64      }
65      public String description() {
66        return "Thermostat on night setting";
67      }
68    }
69    private class ThermostatDay extends Event {
70      public ThermostatDay(long eventTime) {
71        super(eventTime);
72      }
73      public void action() {
74        // Put hardware control code here
75        thermostat = "Day";
76      }
77      public String description() {
78        return "Thermostat on day setting";
79      }
80    }
81    // An example of an action() that inserts a
82    // new one of itself into the event list:
83    private int rings;
84    private class Bell extends Event {
85      public Bell(long eventTime) {
86        super(eventTime);
87      }
88      public void action() {
89        // Ring bell every 2 seconds, rings times:
90        System.out.println("Bing!");
91        if(--rings > 0)
92          addEvent(new Bell(
93            System.currentTimeMillis() + 2000));
```

```
94      }
95      public String description() {
96        return "Ring bell";
97      }
98    }
99    private class Restart extends Event {
100     public Restart(long eventTime) {
101       super(eventTime);
102     }
103     public void action() {
104       long tm = System.currentTimeMillis();
105       // Instead of hard-wiring, you could parse
106       // configuration information from a text
107       // file here:
108       rings = 5;
109       addEvent(new ThermostatNight(tm));
110       addEvent(new LightOn(tm + 1000));
111       addEvent(new LightOff(tm + 2000));
112       addEvent(new WaterOn(tm + 3000));
113       addEvent(new WaterOff(tm + 8000));
114       addEvent(new Bell(tm + 9000));
115       addEvent(new ThermostatDay(tm + 10000));
116       // Can even add a Restart object!
117       addEvent(new Restart(tm + 20000));
118     }
119     public String description() {
120       return "Restarting system";
121     }
122   }
123   public static void main(String[] args) {
124     GreenhouseControls gc =
125       new GreenhouseControls();
126     long tm = System.currentTimeMillis();
127     gc.addEvent(gc.new Restart(tm));
128     gc.run();
129   }
130 } ///:~
```

Exercício 1

Vocês devem agora modelar o universo de StarWars,com base no texto[4]:

- Em 25.000 ABY (Antes da Batalha de Yavin) foi instituído o regime político chamado República Galáctica que foi dissolvido em 19 ABY e que tinha como capital o planeta de Coruscant. Um chefe de estado da república foi Tarsus Valorum que é um Ser macho da espécie humano que nasceu em 1.050 ABY no planeta natal Coruscant. Vários planetas faziam parte da república como: Coruscant de cor metálica no Sistema Coruscant com diâmetro de 12.240 km; Shili de cor azul no Sistema Shili com diâmetro de 23.456 km. Em 24 ABY foi instituída o regime político chamado Confederação de Sistemas Independentes que foi dissolvido em 11 ABY que tinha como capital o planeta de Geonosis. Um chefe de estado da confederação independente foi Conde Dookan que é um Ser macho da espécie humano que se tornou Lorde da ordem Sith, adotando o nome de Darth Tyranus, sendo que nasceu em 102 ABY no planeta natal Serenno. Vários planetas faziam parte da confederação independente como: Geonosis de cor vermelha no Sistema Geonosis com diâmetro de 111.222 km; Serenno de cor preta no Sistema Serenno com diâmetro de 22.222 km; Korriban é um planeta de cor vermelha no Sistema Horuset com diâmetro de 18.984 km e de que tinha como capital a Cidade do Porto. Em 19 ABY foi instituída o regime político chamado Império Galáctico que foi dissolvido em 11 DBY (Depois de Batalha de Yavin) e que tinha como capital o planeta de Coruscant. Um chefe de estado do império foi Papatine que é um Ser macho da espécie humano que se tornou Lorde da ordem Sith adotando o nome de Darth Sidious, sendo que nasceu em 82 ABY no planeta natal Naboo. Vários planetas faziam parte do Império como: Naboo de cor cinza no Sistema Naboo com diâmetro de 32.321 km; Tatooine de cor marrom no Sistema Tatoo com diâmetro de 9.876 km. Um Ser pode aderir a ordem Jedi ou Sith quando apresentasse uma grande concentração de midi-chlorians. Jedi ou Sith são treinados em academias como por exemplo: a Academia Sith Korriban no planeta Korriban que treinava Sith; e a Academia Jedi Coruscant no planeta Coruscant treinava Jedi. Para administrar a academia é formado um único Conselho composto por vários Jedi ou por vários Sith, por exemplo: o Conselho do Templo de Coruscant que administrava a Academia Jedi Coruscant; a Irmandade da Escuridão que administrava a Academia Sith Korriban. Na ordem Jedi deve-se evoluir o percentual de paz interna e podem ou não ter como poderes telepatia, telecinese e persuasão. Conforme seu estágio de treinamento um Jedi podem ser Youngling, Padawan, Cavaleiro ou Mestre. O Youngling ou iniciado na ordem Jedi era um ser criança designada para treinamento em algum Clã sempre treinado por um mestre da ordem Jedi. Um marco importante desse estágio era se o Youngling

[4] O texto é de autoria do Prof. Dr. Enzo Seraphim

já tinha feito a colheita de cristais para seus sabres de luz. Alguns clãs foram: Clã do Urso que tinha como treinador o Mestre Yoda e foi fundado em 4 ABY; Clã Bergruutfaf que tinha como instrutor o Mestre Quarmall, fundado em 45 ABY e extinto em 19 DBY. O Padawan na ordem Jedi deviam usar uma trança quando tinham cabelo, sendo que um marco importante desse estágio era se o Padawan já tinha conhecimento para construir seu sabre de luz. O sabre de luz é feito a partir de um tipo de cristal que gera uma lâmina de uma determinada cor em contato com a manipulação da força. O sabre devia ser montado à mão para alinhar exatamente os cristais irregulares, sendo que o menor desalinhamento provocaria uma explosão ao ativar a lâmina. Os punhos do sabre determinavam o manuseio da lâmina, formando: espada, lança, chicote ou tonfa. Alguns Padawns são: Shaak Ti que é uma Ser fêmea da espécie Togruta, que foi treinada por Anakin Skywalker, sendo que nasceu em 35 ABY no planeta natal Shili; Rey que é uma Ser fêmea da espécie Humana, que foi treinada por Luke Skywalker, sendo que nasceu em 11 DBY no planeta natal Jakku. Cavaleiros na ordem Jedi eram aprovados nos testes Jedi como por exemplo: Luke Skywalker que é um Ser macho da espécie humano, que se tornou Cavaleiro em 1 DBY, sendo que nasceu em 19 ABY no planeta natal Tatooine; Anakin Solo que é um Ser macho da espécie humano, que se tornou Cavaleiro em 26 DBY, sendo que nasceu em 10 DBY no planeta natal Coruscant. O Mestre na ordem Jedi é um Cavaleiro que adquiriu um campo de vidência ou a imortalidade, sendo que alguns exemplos são: Ahsoka Tano que é uma Ser fêmea da espécie de Togruta, que não é imortal e que tem campo de vidência de 10 mil anos-luz, sendo que nasceu em 509 ABY no planeta natal Cevery; Obi-Wan Kenobi que é um Ser macho da espécie humano, que é imortal e que tem campo de vidência de 18 mil anos-luz, sendo que nasceu em 57 ABY no planeta natal Stewjon; Yoda que é um Ser macho da espécie de Yoda, que é imortal e que tem campo de vidência de 100 mil anos-luz, sendo que nasceu em 896 ABY. Na ordem Sith deve-se adotar um novo nome, evoluir o percentual de raiva e podem ou não ter como poderes telepatia, telecinese e persuasão. Conforme seu estágio de treinamento podem ser Aprendizes ou Lordes. O Aprendiz na ordem Sith deve ser treinado por um Lorde, sendo que um marco importante desse estágio era se o Aprendiz já tinha conhecimento para construir seu sabre de luz. Um Lorde da ordem Sith foi Anakin Skywalker que é um Ser macho da espécie humano (ciborgue), que adotou o nome de Darth Vader e que tem campo de vidência de 120mil anos-luz, sendo que nasceu em 42 ABY no planeta natal Tatooine. Em 33 ABY, Darth Sidious elaborou a missão conhecida como Primeira Missão a Ralltiir para acabar com a liderança do sindicato criminal Sol Negro. Darth Maul executou com sucesso essa missão nos planetas Ralltiir do Sistema Ralltiir e Coruscant do sistema Coruscant, sendo que foram envolvidos os humanos: Mighella e Alexi Garyn. Em 19 ABY, Darth Sidious elaborou outra

missão conhecida como Massacre em Mustafar para exterminar os últimos líderes da Confederação de Sistemas Independentes no planeta Mustafar do sistema Mustafar. Darth Vader executou com sucesso essa missão, sendo envolvidos: Denaria Kee da espécie Koorivar, Po Nudo da espécie Aqualish Ualaq, Passel Argente da espécie Koorivar, San Hill da espécie Muun e Tikkes da espécie Quarren. Em 14 DBY, Mestre Yoda elaborou conhecida como Missão a Mos Eisley para investigar a ligação entre contrabandistas e mercenários com o Sith oculto. Jaden Korr e seu mestre da ordem Jedi Kyle Katarn não executaram com sucesso a missão no planeta Tatooine do Sistema Tatoo, sendo envolvidos os humanos Kyle Katarn e Han Solo.

Crie as classes, abstrações, heranças e atributos, nesta atividade não há necessidade de criar métodos além dos métodos de acesso.

Crédito extra: A modelagem deve conter no minimo entre 10 e 15 classes, as situações podem ser modeladas de maneira diferente,o crédito extra sera concedido para as modelagens que mais se aproximarem do gabarito *Dica:* Leia o exercício 2 lá estão algumas dicas de classes que fazem parte da modelagem.

Exercício 2

Vocês devem agora escrever o código que simule uma batalha entre um Mestre Jedi e um Lord Sith, com as seguintes características:

- Vocês irão simular uma batalha, cada personagem desta batalha terá um quantidade de vida, domínio da força e domínio do sabre e uma lista com pelo menos 4 habilidades.Cada uma possui uma prioridade e uma quantidade de dano.
- Cada ação do lutador será um evento: atacar (usando uma habilidade) ou esquivar.
- Esquivar coloca o personagem em um modo onde ele se defende apenas de ataques de sabre mas não afeta ataques com a força.
- Para simplificar a modelagem, as habilidades serão de dois tipos: habilidades com uso da força e habilidades com uso do sabre.Elas terão uma prioridade que deve ser respeitada, se a prioridade de um ataque for menor que a de outro este deve ser executado primeiro.Habilidades de força causam menos dano, mas tem prioridade sobre as habilidades com sabre.Entre os tipos de evento, a prioridade é: esquivar > ataque com força > ataque com sabre.
- A batalha termina quando um dos lutadores não tiver mais vida.

Crédito extra: Altere o código para permitir que o domínio com força ou com sabre de um personagem modifique o dano que a mesma irá causar.