Programação Orientada pelos Objectos

Exame (época de recurso)
2008/07/18

Atenção: A fraude numa prova de avaliação é punida com a reprovação.

Primeiro grupo

As datas de calendário são modeladas pela interface IDate:

```
public interface IDate extends Comparable<IDate>
{
        public int getDay();
        public int getMonth();
        public int getYear();
        public boolean equals(IDate date);
        public boolean isBefore(IDate other);
        public boolean isHoliday();
        public IDate getYesterday();
        public IDate getTomorrow();
        public int workingDaysSince(IDate other);
        public int yearsSince(IDate other);
}
```

Programe a classe DatePortuguese que implementa a interface IDate de acordo com as alíneas adiante indicadas. Considere que esta classe contém os seguintes atributos:

```
static Month[] months =
{
        null,
        new Month("Janeiro", 31),
        new Month("Fevereiro", 28),
        new Month("Março", 31),
        new Month("Abril", 30),
        new Month("Maio", 31),
        new Month("Junho", 30),
        new Month("Julho", 31),
        new Month("Agosto", 31),
        new Month("Setembro", 30),
        new Month("Outubro", 31),
        new Month("Novembro", 30),
        new Month("Dezembro", 31)
};

static private ArrayList<IDate> holidays =
        new ArrayList<IDate>();

static private final char separator = ' ';

private int day, month, year;
```

a) Implemente os métodos da interface IDate que **não estão a sublinhado**. O método isBefore devolve true se this é anterior a other. O método isHoliday devolve true se o dia é um feriado.

b) Implemente os métodos da interface IDate que **estão a sublinhado**. Dado que esta alínea é a mais difícil, sugere-se que a deixe para o final do exame. O método getYesterday devolve o dia anterior a this e o método getTomorrow o dia seguinte. O método workingDaysSince devolve o número de dias que não são feriados desde a data indicada até à data corrente (this). O método yearsSince calcula o número de anos **completos** que decorreram desde other até this. Nestes últimos dois métodos defina uma asserção que garanta que a data passada por parâmetro é anterior a this.

c) Programe os seguintes **construtores**

```
public DatePortuguese (int day, int month, int year)
public DatePortuguese (IDate other)
public DatePortuguese (String dateString)
```

O último destes construtores recebe uma String no formato ano, mês, dia, com separador definido pela constante estática separator.

d) Programe os métodos **estáticos** seguintes:

```
public static Iterable<IDate> getHolidays()
public static void loadHolidays(Scanner in)
public static boolean isLeapYear(int year)
public static int getMonthLength(int month, int year)
```

O método loadHolidays lê linhas com todos os feriados existentes no calendário. Cada linha tem um formato igual ao descrito na alínea anterior. O método isLeapYear devolve true se o ano é bissexto (divisível por 4, mas não divisível por 400). O método getMonthLength devolve a dimensão em dias do mês indicado, no ano indicado.

Segundo grupo

a) Programe a classe Friend (amigo) com a seguinte estrutura:

```
public class Friend {
        private String name;
        private DatePortuguese birthday;
        public Friend(String name, IDate birthday)
        public Friend(Friend other) …
        public String getName() …
        public IDate getBirthday() …
        public boolean isAniversary(IDate date) …
        public int getAge(IDate date)
        public boolean equals(Object other)
        public String toString()
}
```

O método isAniversary indica se na data passada por parâmetro se comemora o aniversário do amigo. O método getAge dá a idade do amigo na data passada por parâmetro, calculada da forma que todos nós conhecemos. O método equals só dá true se quer o nome, quer a data de nascimento forem iguais. O valor retornado por toString deve ter o formato que aparece na janela representada na figura nas linhas que contêm o nome de um amigo (ex: João, nascido a …).

b) Programe a classe Agenda de aniversários de amigos com a seguinte estrutura:

```
public class Agenda implements Iterator<IDate>
{
        private ArrayList<Friend> friends;
    private IDate startDate, stopDate;

        public Agenda(IDate start, IDate stop)
        public void setLimits(IDate start, IDate stop)
        public boolean addFriend(Friend friend)
        public boolean removeFriend(Friend friend)
        public Friend getFriend(String name)
        public IDate getBirthday(String name)
        public Iterable<Friend> allFriends()
        public Iterable<Friend>
                        getAnniversariants(IDate aDate)
}
```

As datas startDate e stopDate correspondem aos limites da agenda, que podem ser alterados através do método setLimits. Os métodos getFriend e getBirthday devolvem null se não encontrarem um amigo com o nome dado. Os métodos allFriends e getAnniversariants devolvem uma colecção iterável de amigos (todos, no 1º caso, ou só os que comemoram o aniversário na data dada, no 2º).

c) Programe os métodos hasNext e next da interface Iterator<IDate> que a classe Agenda implementa (não é preciso implementar o método remove. A primeira chamada à função next deve devolver a primeira data da agenda e assim sucessivamente até chegar à última. Recorda-se que a interface define os seguintes métodos:

```
public boolean hasNext()
public IDate next()
```

Terceiro grupo

Desenhe um diagrama de classes em UML relativo ao proposto nos grupos 1 e 2. Represente os atributos, relações de herança, implementação e dependência. Não represente as operações.

Quarto grupo

a) Crie na classe Agenda o método público Iterable<String> contents() em que cada linha contém informação sobre as datas contidas na agenda, indicando se são ou não feriados, tal como representado na área central da janela ao lado para o caso do 1º de Maio. Se numa dada data se comemora o aniversário de alguém, então é adicionada uma linha por cada um desses aniversários, também com o formato representado na janela ao lado. *(Sugestão: use o método getAnniversariants definido na classe Agenda).*

b) Crie a classe AgendaPanel que herda de JPanel, com um formato semelhante ao representado na figura, ou seja, no topo dois campos de entrada de dados (JTextField), com as respectivas legendas (JLabel), para colocar as datas inicial e final, ao meio o conteúdo da agenda entre as datas especificadas (JTextArea) e em baixo um botão (JButton). O conteúdo da agenda, no intervalo de datas indicado, obtido a partir da chamada ao método contents da alínea a), só é mostrado quando se carrega no botão.

c) Escreva o programa principal que carrega o ficheiro de feriados (com tratamento de excepções adequado) e mostra o painel criado na alínea anterior.