Auxiliar 3

Repaso, parte 1

Profesor: Alexandre Bergel
Auxiliares: Javier Espinoza
Eduardo Riveros
Fecha: 28 de Agosto del 2017

1. HashMaps / Hashtables

Los *Hashmap* (O *Hashtable como seguramente vieron en clases*) son estructuras de datos similares a los *diccionarios* de Python, que permiten guardar **objetos** y recuperarlos con **llaves** (como por ejemplo, cadenas de texto) en tiempo *constante*. No entraremos en detalles de su implementación porque se escapa del objetivo del curso, pero sí discutiremos su interfaz.

Para crear un nuevo *Hashtable*, es necesario instanciarlo como cualquier otra clase, pero además hay que explicitar los *generic types* (materia que veremos en unos meses) usados como llave y como valor.

Ejemplificaremos su uso con un pequeño programa que entrega el horóscopo:

Código 1. HashMap.java

```
class TiaYoli {
  public static void main(String[] args) {
    // Creamos un hashtable
    Map<String, String> horoscopo  = new Hashtable<String, String>(); // El valor será el
horóscopo para hoy.
    // Colocamos algunos valores (Son libres de colocar sus predicciones para el semestre)
    horoscopo.put("Aries", "Tu tarea funcionará y sí sabrás por qué.");
    horoscopo.put("Tauro", "Tu promedio en metodologías corre peligro. No faltes a las
auxiliares");
    horoscopo.put("Leo", "El mayor code smell de tu vida está a punto de presentarse frente a
ti");
    // Mostrar un horóscopo
    System.out.println(horoscopo.get("Leo"));
    // Borrar un horóscopo porque no nos gustó
    horoscopo.remove("Leo");
    // Ahora get debería devolver null.
    System.out.println(horoscopo.get("Leo"));
  }
}
```

2. Herencia y Lookups.

Basándose en las siguientes clases, determinar los output producidos por los respectivos *main*.

Código 1. Doggo.java

```
class Doggo {
  public void wow() {
    System.out.println("im a doggo wow");
  }
}


class Pupper extends Doggo {
  public void wow() {
    System.out.println("im a pupper wow wow");
  }
}


class Me {
  public void adopt(Doggo e) {
    System.out.println("i adopted a doggo");
  }

  public void adopt(Pupper e) {
    System.out.println("i adopted a pupper");
  }
}


class Main {
  public static void main(String[] argv) {
    Me me = new Me();
    Doggo doggo1 = new Pupper();
    Pupper doggo2 = new Pupper();
    me.adopt(doggo1);
    me.adopt(doggo2);
    me.adopt(new Pupper());
    doggo1.wow();
    doggo2.wow();
    (new Doggo()).wow();
  }
}
```

Código 2. Fogs.java

```
abstract class AbstractFogs {

  public AbstractFogs() {
    System.out.println("small startle");
  }

  public AbstractFogs(int i) {
    this();
    System.out.println("big startle " + i);
  }

}

class PresidentFogs extends AbstractFogs {
  public PresidentFogs() {
    this(99);
    System.out.println("im presiden");
  }

  public PresidentFogs(int i) {
    super(i);
    System.out.println("presiden please");
  }
}

public class Fogs {
  public static void main(String[] args) {
    new PresidentFogs();
  }
}
```

Código 3. Frases.java

```
class FraseCortante {
  public String adornar(String frase) {
    return frase + ".";
  }

  public String reirMucho(String frase) {
    return this.adornar(frase) + " xd";
  }
}

class FraseSuavizada extends FraseCortante {
  public String adornar(String frase) {
    return frase + " jaja";
  }
}

public class Frases {
  public static void main(String[] args) {
    System.out.println((new FraseSuavizada()).reirMucho("Nunca le entiendo al auxiliar cuando
explica"));
  }
}
```

3. Difusión en el Centro de Alumnos

En el **Centro de Alumnos del Departamento de Ciencias de la Computación**, nos demoramos mucho en difundir informaciones debido a que manejamos muchos canales de comunicación.

Actualmente, existen **tres** tipos de informaciones que se difunden prácticamente todos los días:

1. **Afiches**, que son imágenes que contienen en ellas mismas toda la información de los eventos que promocionan.
2. **Videos**, que son recopilaciones audiovisuales de eventos de la semana y se difunden sólo en algunos medios.
3. **Avisos,** que son textos cortos que dan a conocer algún tema de interés público..

La información anterior se reparte en los siguientes medios:

1. **Blog,** hecho para mostrar **afiches**, **videos** y **avisos**.
2. **Página** en **Facebook**, que muestra **afiches** y **videos**.
3. **Pantalla** cerca de la sala de alumnos que muestra solamente **afiches**.
4. **Canal** de **Telegram**, que muestra **afiches** y **avisos**.
5. **Usuario** en **Twitter**, que muestra solamente **avisos**.
6. **Comunidad** en **Ucursos**, que muestra **videos** y **avisos**.

En estos momentos es muy difícil difundir manualmente, medio por medio, toda la información (Además que nos olvidamos qué va donde y muchas veces nos equivocamos al publicar). Dado lo anterior, calculamos que nos demoramos aproximadamente una hora en enviar todo a los canales al publicarla manualmente.

En un intento de optimizar nuestro tiempo, y considerando que es necesario informar muchas cosas este semestre (en especial eventos como las actividades de movilización del miércoles o la **Bienvenida Mechona de este viernes a las 14:00 hrs**) intentamos crear un sistema que automatizará todo el proceso de publicación y difusión de información. Lamentablemente, la falta de tiempo nos ha complicado para enfocarnos en su desarrollo, y sólo tenemos hechos los tests unitarios (en el paquete *tests)*, propiedades de los objetos que representan los tipos de información (en el paquete *content*) y una API antigua (en el paquete *cadcc*) que se encarga de realizar efectivamente las publicaciones en cada medio.

Con lo entregado en el adjunto a esta auxiliar, la idea es discutir entre todos cuál es la mejor forma de implementar una solución.