EJERCICIO RESUELTO
Módulo: Programación

Operaciones sobre Db4o

Descripción:

Supongamos que tenemos las siguientes clases implementadas:

La clase Player

```
public class Player {

    protected String name;
    protected int squadNumber;
    protected float battingAverage;
    protected Team team;

    public Player(String name, int squadNumber,
        float battingAverage){
        this.name = name;
        this.squadNumber = squadNumber;
        this.battingAverage = battingAverage;
    }

    public void setName(String n){this.name = n;}
    public String getName(){return this.name;}

    public void setSquadNumber(int s){this.squadNumber = s;}
    public int getSquadNumber(){return this.squadNumber;}

    public void setBattingAverage(final float b) {
        this.battingAverage = b; }
    public float getBattingAverage(){
        return this.battingAverage;}
```

```
    public void setTeam(Team t) {this.team = t;}
    public Team getTeam() {return this.team;}
```

```
    public String toString() {
        return name + ":" + battingAverage;
    }
}
```

La clase Pitcher

```
public class Pitcher extends Player{
    private int wins;

    public Pitcher(String name, int squadNumber,
                   float battingAverage, int wins) {
        super(name,squadNumber,battingAverage);
        this.wins = wins;
    }

    public void setWins(final int w){this.wins = w;}
    public int getWins() {return this.wins;}

    public String toString() {
        return name + ":" + battingAverage + ", " + wins;
    }
}
```

La clase Team

```
import java.util.List;
import java.util.ArrayList;

public class Team {

    private String name;
    private String city;
    private int won;
```

```
    private int lost;
    private List players;

    public Team(String name, String city,
                int won, int lost){
        this.name = name;
        this.city = city;
        this.won = won;
        this.lost = lost;
        this.players = new ArrayList();
    }

    public void addPlayer(Player p) {
        players.add(p);
    }

    public void setName(String n){this.name = n;}
    public String getName(){return this.name;}

    public void setStadium(String c){this.city = c;}
    public String getCity(){return this.city;}

    public void setPlayers(List p){players = p;}
    public List getPlayers(){return players;}

    public void setWon(int w) {this.won = w;}
    public int getWon(){return this.won;}

    public void setLost(int l) {this.lost = l;}
    public int getLost() {return this.lost;}

    public String toString() {
        return name;
    }
}
```

Primero, creamos algunos datos de prueba con los que trabajar:

```
// Create Players
Player p1 = new Player("Barry Bonds", 25, 0.362f);
Player p2 = new Player("Marquis Grissom", 9, 0.279f);
Player p3 = new Player("Ray Durham", 5, 0.282f);
Player p4 = new Player("Adrian Beltre", 29, 0.334f);
Player p5 = new Player("Cesar Izturis", 3, 0.288f);
Player p6 = new Player("Shawn Green", 15, 0.266f);

// Create Pitchers
Player p7 = new Pitcher("Kirk Rueter",46, 0.131f, 9);
Player p8 = new Pitcher("Kazuhisa Ishii",17, 0.127f, 13);

// Create Teams
Team t1 = new Team("Giants", "San Francisco", 91, 71);
Team t2 = new Team("Dodgers", "Los Angeles", 93, 69);

// Add Players to Teams
t1.addPlayer(p1); p1.setTeam(t1);
t1.addPlayer(p2); p2.setTeam(t1);
t1.addPlayer(p3); p3.setTeam(t1);
t2.addPlayer(p4); p4.setTeam(t2);
t2.addPlayer(p5); p5.setTeam(t2);
t2.addPlayer(p6); p6.setTeam(t2);

// Add Pitchers to Teams
t1.addPlayer(p7); p7.setTeam(t1);
t2.addPlayer(p8); p8.setTeam(t2);
```

Deseamos realizar las siguientes operaciones:

1 – Almacenar un objeto en una base de datos dB4o
2 – Consultar un dato concreto de la base de datos.
3 – Obtener todos los objetos player que se han almacenado
4 – Actualizar y borrar.

Objetivos:

- Comprender las BD Orientadas a Objeto.
- Conocer las características de una BDOO.
- Explotar una BDOO Db4o haciendo inserciones, consultas simples y consultas resumen

Recursos:

- Acceso a Internet.
- Netbeans o Eclipse
- Db4o ObjectDatabase

Resolución:

1

Un objeto `Team` se puede almacenar con una sóla línea de código:

```
db.set(t1);
```

Donde `db` es una referencia a un objeto `ObjectContainer`, que se haya creado abriendo un fichero de base de datos, de esta forma:

```
ObjectContainer db = Db4o.openFile(filename);
```

Una base de datos **db4o** es un único fichero con una extensión `.yap`, y se utiliza su método `set` para almacenar objetos.

Observe que está linea almacena el objeto `Team` **y** su colección de objetos `Player`. Se puede probar esto recuperando uno de esos objetos `Player`.

2

El siguiente código lista todos los objetos `Player` que sean iguales al objeto de ejemplo; sólo debería haber un resultado. Los resultados se obtienen como un `ObjectSet` llamando al método `get` de `ObjectContainer`.

```
Player examplePlayer = new Player("Barry Bonds",0,0f);
ObjectSet result=db.get(examplePlayer);


System.out.println(result.size());
while(result.hasNext()) {
    System.out.println(result.next());
}
```

3

Se pueden obtener **todos** los objetos `Player` que se hayan almacenado creando un objeto de ejemplo vacío (todos son campos son `null` o `0`), de esta forma:

```
Player examplePlayer = new Player(null,0,0f);
ObjectSet result=db.get(examplePlayer);
```

```
System.out.println(result.size());
while(result.hasNext()) {
    System.out.println(result.next());
}
```

4 –Actualizar y borrar

La actualización de objetos se pude conseguir utilizando una combinación de las técnicas anteriores. El siguiente código asume que sólo se ha encontrado una correspondencia, y el objeto encontrado se fuerza a `Player` para poder modificar sus atributos:

```
Player examplePlayer = new Player("Shawn Green",0,0f);
ObjectSet result = db.get(examplePlayer);
Player p = (Player) result.next();
p.setBattingAverage(0.299f);
db.set(p);
```

De forma similar se pueden borrar objetos de la base de datos:

```
Player examplePlayer = new Player("Ray Durham",0,0f);
ObjectSet result = db.get(examplePlayer);
Player p = (Player) result.next();
db.delete(p);
```