Informe Programación 2

Integrantes:
Lucas Rueda
Ornella Albarracin Gioia

Docentes:
Martin Pustilnik
Ximena Ebertz

Introducción

El propósito de este trabajo práctico es, de manera grupal, llevar a cabo el análisis de una problemática a resolver. A partir de ello, se presentará a lo largo del informe el diseño y desarrollo de la solución elegida, indicando los conceptos orientados a objetos que se utilizaron; tales como herencia, polimorfismo, y por otro lado, sobreescritura, sobrecarga e interfaces.

Descripción

Implementamos en la Clase AlbumDelMundial la interfaz dada por la cátedra IAlbumDelMundial para traer los métodos y desarrollarlos durante el trabajo. Además, se implementó una clase abstracta Álbum para distribuirla en sus clases hijas (Tradicional, Web, Extendido) determinando el concepto de herencia.
Por otra parte, la Sobreescritura con @Override se empleó en los métodos que necesitaban ser específicos y únicos según cada clase (como darPremio, estaCompletoAlbum, equals, toString y hashcode).
Y por último, se aplicó Polimorfismo de manera que se logre trabajar con distintos tipos de Objetos sobre un mismo método, por ejemplo, al ingresar el dni de un participante o para el método privado mismoOMenorValor() que recibe cualquiera de los 2 tipos de Figuritas para calcular su valor.

Tecnologias de Java:

StringBuilder: Se aplicó en el listadoDeGanadores y para el toString de las clases.

Iterator: En la clase Participante, el método pegarFiguritas() utiliza el iterator para recorrer la colección de figuritas del participante y si es necesario removerlas y finalmente actualizar la colección.

ForEach: en general los usamos para recorrer las colecciones o arrayList, de las clases Participantes, Pais y Album.

Especificación

Class AlbumDelMundial implement IAlbumDelMundial:

Atributos:

Random random: Variable de número random.
Map <Integer, Participante> _participantes: Map de Participantes registrados.
Fabrica fabrica: Variable de tipo Fabrica.

Métodos:

AlbumDelMundial() //Constructor

public String toString()

- **public void registrarParticipante(String nombre, int dni, String tipo){**

Valida que los datos ingresados como parámetro sean válidos.
Luego, se inicializa un nuevo Participante con los datos parametrizados y lo agregamos a _participantes.put(dni, participante).
}

- **public void comprarFiguritas(int dni){**

Verifica que el dni esté asociado a un participante en _participantes.
Agrega 4 figuritas tradicionales de manera aleatoria al participante.

}

- **public void comprarFiguritasTop10(int dni){**

Verifica que el dni sea válido.
Le asigna 4 figuritas Top10 aleatorias al participante.
}

- **public void comprarFiguritasConCodigoPromocional(int dni){**

Verifica que el dni sea válido y que el participante tenga disponible el código promocional, de lo contrario no puede volver a usarlo.
Le agrega 4 figuritas tradicionales sin costo al participante si tiene un álbum Web.
}

- **public List <Figurita> pegarFiguritas(int dni){**

Busca al participante que tenga el dni llegado como parámetro y le pega todas las figuritas de la colección que no estén repetidas o que aún no se hayan pegado en el álbum adquirido.

Finalmente retorna una lista con las figuritas que fueron pegadas correctamente.
}

- **public boolean llenoAlbum (int dni){**
  Retorna true si el dni ingresado como parámetro pertenece a algún participante en _participantes que tenga su álbum completo y false en caso contrario.
  }

- **public String aplicarSorteoInstantaneo (int dni){**
  Verifica que el dni esté asociado a un participante en _participantes y si dicho participante tiene un álbum tradicional o extendido.
  De manera aleatoria se sortea un premio al participante.
  }

- **public int buscarFiguritaRepetida(int dni){**
  Valida que el dni ingresado esté registrado y devuelve el ID de una figurita repetida que contenga el participante en su colección.
  }

- **public boolean intercambiar (int dni, int codFigurita){**

Verifica que el dni pertenezca a un participante y el id pertenezca una figurita en colección.
Recorre todos los participantes hasta encontrar alguno que tenga una figurita repetida del mismo o menor valor para intercambiar. De esta manera, realiza el intercambio entre ambos y retorna true.

}

- **public boolean intercambiarFiguritaRepetida(int dni){**
  Valida el dni ingresado, y retorna true si se pudo intercambiar y false en caso contrario;
  }

- **public String darNombre(int dni){**
  Valida el dni y retorna el nombre del participante.
  }
- **public String darPremio(int dni){**
  Retorna un String con el premio que le corresponde al participante sólo si completa su Álbum de lo contrario se arroja una excepción.
  }

- **public String listadoDeGanadores( ){**
  Retorna un String de los participantes que lograron completar su álbum y el premio que les corresponde según el tipo de álbum.
  }

- **public List <String>participantesQueCompletaronElPais(String nombrePais){**
  Retorna una lista con todos los participantes que completaron su álbum con el país ingresado como parámetro.
  }

IREP

Map <Integer, Participante> _participantes

1) Para todo participante en _participantes.claves():
   participante == _participantes.obtener(participante).getDni().

Class Album abstract

Atributos:

static int count: Variable de clase incremental.
String _tipo: Variable tipo de álbum.
int _id: Variable que contiene el ID del álbum.
Map <String, Pais> _equipos: Map con los equipos clasificados para el Mundial.

Métodos:

Album(String tipo) //Constructor

- **public String toString()**

- **public boolean sePegoFigurita(Figurita figurita){**
  Recibe una figurita, retorna true si en algún país en _equipos ya contiene la figurita, y false en caso contrario.
  }

- **public void pegarFigurita (Figurita figurita){**
  Si la figurita que le llega como parámetro no se encuentra en _equipos la pega.
  }

- **public boolean estaCompletoAlbum(){**
  Retorna true si para cada pais en _equipos, su método estaCompletoPais() es verdadero. Y false en caso contrario.

- **public boolean completoPais(String nombrePais){**
  Retorna true si para la clave nombrePais en _equipos, su valor pais.estaCompletoPais().
  }

- public void cargarPaises(Map<String, Pais> _paises){
  Inicializa el map _equipos con los paises que le llegan por parámetro
  }

- **public abstract String darPremio()** {
  Este método es abstracto porque cada uno de sus hijos lo va a implementar de forma diferente para retornar su premio particular.
  }

IREP

_tipo
1) El parámetro ingresado debe ser != null y ==“Tradicional”) || == “Extendido” || == “Web”

_id
1) _id == count y _id> 0 && _id != null

count
1) count >= 0

Map<String, Pais>_equipos
1) 0 <= _equipos.size() <= 32
2) Para toda clave c en _equipos c == _equipos.get(c).get_nombre()

Class Web extend Album

Atributos:

boolean _codigoProm: Variable que indica si el Participante ya utilizó su código promocional.

Métodos:

Web(String tipo) //Constructor

- **public boolean tienecodigoDisponible(){**
  retorna _codigoProm.
  }

- public void usarCodigo(){
  Vuelve a false la variable _codigoProm para que no pueda volver a usarse.

  }

- **public String darPremio(){**
  Retorna un String indicando el premio perteneciente al Álbum web.
  }

IREP

boolean _codigoProm
1) _codigoProm | | !_codigoProm

Class Tradicional extends Album

Atributos:
boolean _solicitoSorteo: Variable que indica si el Participante ya solicitó su sorteo.

Métodos:

Tradicional(String tipo) //Constructor

- public String darPremio(){
  Retorna un String indicando el premio perteneciente al Álbum Tradicional.
  }

- public boolean solicitoSorteo{
  retorna _solicitoSorteo.
  }

- public void sorteoRealizado{
  Vuelve a true _solicitoSorteo para indicar que el participante no solicite nuevamente un premio.
  }

IREP
boolean _solicitoSorteo:
_solicitoSorteo | | !_solicitoSorteo.

Class Extendido extends Tradicional

Atributos:
List < Ftop10> _figuritasTop10;

- public String darPremio(){
  Retorna un String indicando el premio perteneciente al Álbum Extendido.
  }

- public boolean sePegoFiguritaTop10(Ftop10 figurita){
  Retorna true si la figurita está contenida en _figuritasTop10.
  }

- **public void pegarFiguritasTop10(Ftop10 figurita){**
  Si la figurita pasada por parámetro no está contenida y no es null, la agrega a _figuritasTop10

IREP

List <Ftop10 > _figuritasTop10

1) Para toda figurita en _figuritasTop10, figurita1 != figurita2
2) 0 <= _figuritasTop10.size <= 20
3) tipo ArrayList<Ftop10>()

Class Participante

Atributos:

int _dni: Variable que contiene el número de dni del participante.
String _nombre: Variable que indica el nombre del participante.
Album _album: Variable de clase Album.
List<Figurita> _coleccionFiguritas: Lista que contiene todas las figuritas que va adquiriendo el participante.

Métodos:

Participante(Integer dni, String nombre, Album album) // constructor

- **boolean equals(Object obj){**
  Si el obj no es null o de otro tipo, lo castea al tipo Participante y compara los atributos con los del obj
  }

- **String toString(){**
  retorna " - $_dni $_nombre : $darPremio()) + "\n" + _album.toString()
  }

- **public void agregarFiguritaAColeccion(List<FIGURITA> sobre){**
  recorre la list pasada por parámetro y agrega los elementos a _coleccionFiguritas
  }

- **boolean tieneCodigoDisponible(){**
  Si tiene un álbum del tipo Web consulta al _album el estado de _codigoProm
  }

- **boolean tieneSorteoDisponible(){**
  Si tiene un álbum del tipo Tradicional consulta a _album el estado de _solicitoSorteo
  }
- **void usarCodigo(){**
  Cuando se llama a este metodo cambia el estado de _codigoProm para que el participante no pueda solicitar nuevamente la promoción de figuritas
  }

- **void usarSorteo(){**
  Cuando se llama a este metodo cambia el estado de _solicitoSorteo para que el participante no pueda solicitar nuevamente el Sorteo Inmediato
  }

- **public List<String> pegarFigurita(){**
  Recorre la _coleccionFiguritas por medio de Iterator<Figurita> las va agregando al album y removiendo de la colección y agregando la figurita.toString() a una List<String>.
  Retorna una lista de string con todas las figuritas en _coleccionFiguritas que fueron correctamente pegadas.
  }

- **public boolean albumCompleto(){**
  Retorna true si _album está completo y false en caso contrario.
  }

- **int traerID_figuritaRepetida(){**
  retorna el id de la primera figurita repetida que encuentra, si _coleccionFiguritas esta vacio o no encuentra ninguna figurita retorna -1.
  }

- **public boolean completoPais(String nombrePais){**
  Retorna true si el pais en el album esta completo
}

- **boolean poseeFigurita(int codFigurita){**
  retorna true si el codFigurita coincide con el id de alguna figurita en la colección.
  }

- **Figurita traerFigurita(int codFigurita){**
  retorna la figurita con el id que coincida con el codFigurita
  }

- **public void intercambiar(Figurita figurita, Figurita nuevaFigurita){**
  Elimina de _coleccionFiguritas figurita y agrega a _coleccionFiguritas nuevaFigurita.

}

IREP

_dni

_dni != null && _dni > 0.

_nombre

_nombre != null && _nombre.length() > 0

_album

_album != null &&
_album instanceof “Tradicional” || “Extendido” || “Web”

_coleccionFiguritas

tipo ArrayList<Figurita>, puede contener elementos repetidos

Class Figurita

Atributos:

int _id: Variable que indica el ID de la Figurita.
String _tipo: Variable que contiene el tipo de la figurita.
int _numJugador: Variable que representa el número del jugador.
String _nombrePais: Variable que guarda el nombre del país del jugador.

Métodos:

- **Figurita (String tipo, int numJugador, String nombrePais)** //Constructor

- **toString(){**
  retorna un string en formato “$_nombrePais - Jugador: _numJugador”
  }

- int hashCode{
  retorna la ecuacion: (_nombrePais.length() + _numJugador + _tipo.length()) * 7 + _numJugador
  }
- **boolean equals(Object obj)**{
  Recibe un Objeto, chequea que no sea Null o de otro tipo de dato que no sea figurita. Lo castea al Tipo Figurita y compara sus atributos con los del obj
  }

IREP

_id

0 < _id && _id!=null, se define con el hashCode

_tipo

_tipo != null && (_tipo.equals(“Tradicional”) | | _tipo.equals(“Top10”))

_numJugador
0 =< _numJugador. < 12

_nombrePais
_nombrePais != null && _nombreJPais.length() > 0

Class FTop10 extends Figurita

Atributos:
String _paisSede: variable que contiene el país sede.
String _balon: variable que contiene el tipo de balón.

Métodos:

- FTop10 (String nombrePais, String paisSede, String balón) // constructor

- String toString(){
  Al toString del padre le agrega _paisSede y el _balon
  }

- int hashCode(){
  Al hashCode del padre le suma _balon.length() * _paisSede.length() * 10 para una mayor diferenciación con las Figuritas tradicionales
  }

- boolean equals(Object obj) {
  Si el Object es del tipo Ftop10 y no es null, lo castea al tipo Ftop10, compara los atributos y con el equals del padre le pasamos por parámetro el obj casteado al tipo Figurita. Si se cumplen todas esas condiciones retorna true.

}

IREP

_paisSede
_paisSede != null && _paisSede.length() > 0
_ balón
_balon != null && _balon.length() > 0
_balon.equals("oro") | | _balon.equals("plata")

Class Pais

Atributos:
String _nombre: Variable con el nombre del país.
int _ranking: Variable que indica el ranking del país.

List<Figurita> _figuritasDeJugadores: Lista de la figuritas de los jugadores.

Métodos:

Pais (String nombre, Integer ranking){} // constructor

- **toString()**

- **public boolean sePegoFigurita(Figurita figurita){**
  Retorna true si la figurita que le llega como parámetro está contenida en _figuritasDeJugadores o false en caso contrario.
  }

- **public void pegarFiguritas (Figurita figurita){**
  Agrega la figurita pasada por parámetro a la lista de figuritas si esta no es nula o ya se encuentra agregada.
  }

- **public boolean estaCompleto(){**
  Retorna true si _figuritasDeJugadores ya contiene los 12 jugadores pertenecientes al país, y false en caso contrario.
  }

IREP

_nombre
_nombre != null & _nombre.length > 0.

_ranking
_rankink != null && _ranking > 0.

List <figurita>_figuritasDeJugadores

1) 0 <= _figuritasDeJugadores.size() <= 12

2) Pegar una figurita implica agregarla a la List de _figuritasDeJugadores

3) no puede haber una figurita repetida

Métodos incluidos dentro de la Clase Fabrica

- public Map<String, Pais> get_paises() {
  Retorna el Map armado con los países clasificados en generarPaises()..

}

- **private Map<String, Pais> generarPaises(){**
  Arma un Map con los países participantes y su ranking del Mundial
  }

- **Album crearAlbumWeb(){**
  Retorna un Album Web.
  }

- **Album crearAlbumExtendido(){**
  Retorna un Album Extendido.
  }

- **Album crearAlbumTradicional(){**
  Retorna un Album Tradicional.
  }

- **public String sortearPremio(int numSorteo){**
  Retorna aleatoriamente un premio que se encuentre dentro del Array de premiosInstantaneos.
  }

- **List<Figurita> generarSobre(int cantFigus){**
  Retorna una Lista de Figuritas tradicionales según la cantFigus que se indican por parámetro generadas aleatoriamente.
  }

- **List<Figurita> generarSobreTop10(int cantFigus){**
  Retorna una Lista de Figuritas Top10 según la cantFigus que se indican por parámetro generadas aleatoriamente.
  }

- **public int valorBase(Figurita fig){**
  Retorna el cálculo de valorBase de la Figurita llegada como parámetro.
  }

Complejidad de buscarFIguritaRepetida(int dni)

```
public int traerID_figuritaRepetida() {                                  operaciones

        if (_coleccionFiguritas.size() == 0) {                                1
                return -1;                                                    1
        }

        return _coleccionFiguritas.get(0).get_id();                           2
}                                                                        O(4) = O(1)
```

```
public int buscarFiguritaRepetida(int dni) {                    operaciones

 if (!_participantes.containsKey(dni)) {                              2
        throw new RuntimeException("No se encuentra registrado");     1
                }

 Participante participante = _participantes.get(dni);                 2

 return participante.traerID_figuritaRepetida();                      2
}
```

Operaciones

O(2) + O(1) + O(2) + O(2)

O(7)

O(1)

Complejidad: O(1)

Se establece finalmente que la complejidad de buscarFiguritaRepetida(int dni) es de complejidad O(1).

Dificultades y soluciones

Las dificultades más significativas que se presentaron a lo largo del trabajo, fue primeramente amoldarnos a otro diseño y poder encontrar otras soluciones en comparación con las que ya se habían planteado en el diagrama de clases y la especificación de Tads.
También, resultó difícil la implementación del método intercambiar( ) puesto que, debíamos validar y tener en cuenta diversas situaciones como por ejemplo, no intercambiar figuritas iguales entre los participantes sino del mismo o menor valor, validar si son de diferente tipo (Figurita Tradicional o Figurita Top10) ya que según sea la instancia de la Figurita se debía ejecutar otro código.

Conclusión

En conclusión, el desarrollo del trabajo práctico nos permitió colaborar en equipo, plantear ideas, debatir soluciones y decidir en conjunto para resolverlo.
En relación con la utilización de nuevos conceptos sobre Programación Orientada a Objetos y herramientas de Java, nos permitieron adquirir más conocimientos que podremos aplicar más adelante para seguir desarrollando lógicas más complejas.