Prueba 3 de Evaluación Continua

Análisis y Diseño de Software (2012/2013)

Contesta AL DORSO, usa hojas adicionales si fuese necesario

Apellidos:
Nombre:

Apartado 1 (4 puntos)

Completa el siguiente código para que la salida del programa sea la dada abajo. Cada carta queda identificada por su palo y su número, y tiene otro atributo más que identifica el juego en el que se va a utilizar. *El valor de cada carta lo calcula el propio juego para el que haya sido creada la carta, que podrá tener o no en cuenta los atributos palo y número de la carta.* Para añadir las declaraciones e inicialización de las variable locales necesarias en el método **`main()`** debe prestarse atención a la ordenación y tratamiento de duplicados en la salida dada. Recuérdese que todos los valores de tipos enumerados son comparables según el orden establecido por su posición dentro de la definición del tipo enumerado

```
package prueba3;       // primera línea deber ir en todos los archivos *.java
import java.util.*;    // segunda línea irá en todos los archivos que sea necesario

public enum Palo {OROS, COPAS, ESPADAS, BASTOS};

public interface JuegoCartas {
       /**  cada juego de cartas debe definir el valor de cada carta */
       public Integer valor(Carta carta);
       /**   cada juego implementa la estrategia básica para que el jugador que tiene el turno
             pueda elegir una carta de las que tiene en su mano conociendo las cartas
             que están visibles sobre la mesa jugadas anteriormente por otros jugadores */
       public Carta elegirCarta(Set<Carta> mesa, Set<Carta> mano);
}
public class Carta                                                  {  // completar si fuese necesario
       private Palo palo;
       private Integer numero;
       private JuegoCartas juego;

       public Carta(Palo p, Integer num, JuegoCartas j) {this.palo = p; this.numero = num;  this.juego = j;}
       public Palo getPalo() { return palo; }
       public Integer getNumero() { return numero; }
       public String toString() { return getPalo()+":"+getNumero(); }

       // El valor de cada carta VENDRÁ DADO POR EL JUEGO en que se esté usando la carta
       public Integer valor() {         // completar este método

       }

       // añadir los métodos necesarios para completar la clase Carta

}
// añadir la definición completa de la clase ComparadorPorValorDeCarta que se utiliza más abajo

/* No debes definir la clase JuegoSimple. Asume que ya existe y es una implementación correcta de la
  interfaz JuegoCartas, con este método que devuelve el numero del carta como valor de la carta en el juego:
       public Integer valor(Carta carta)  {  return carta.getNumero(); } */

public class Main1 {
      public static void main(String[] args) {
        // añadir declaraciones e incializaciones de variables locales mazo1, mazo2 y mazo3

        JuegoCartas j = new JuegoSimple();  // rellenamos mazo1 con algunas cartas variadas
        mazo1.add( new Carta(Palo.ESPADAS,6,j) );          mazo1.add( new Carta(Palo.ESPADAS,1,j) );
        mazo1.add( new Carta(Palo.OROS,6,j) );             mazo1.add( new Carta(Palo.OROS,1,j) );
        mazo1.add( new Carta(Palo.COPAS,12,j) );           mazo1.add( new Carta(Palo.OROS,6,j) );
        Collections.sort(mazo1, new ComparadorPorValorDeCarta()); // mazo1 queda ordenado
        System.out.println(mazo1);
        for (Carta c : mazo1) {
            mazo2.get(c.getPalo()).add(c);
            if (mazo3.containsKey(c)) mazo3.put(c, mazo3.get(c) + 1);
            else mazo3.put(c,1);
        }
        System.out.println(mazo2);
        System.out.println(mazo3);
     }
 }
```

Salida:

```
[ESPADAS:1, OROS:1, ESPADAS:6, OROS:6, OROS:6, COPAS:12]
{OROS=[OROS:1, OROS:6, OROS:6], COPAS=[COPAS:12], ESPADAS=[ESPADAS:1, ESPADAS:6], BASTOS=[]}
{ESPADAS:6=1, OROS:6=2, ESPADAS:1=1, COPAS:12=1, OROS:1=1}
```

Prueba 3 de Evaluación Continua

Análisis y Diseño de Software (2012/2013)

Contesta AL DORSO, usa hojas adicionales si fuese necesario

Apellidos:
Nombre:

Apartado 2 (3.5 puntos)

Partiendo del código del apartado anterior, hemos cambiado la interfaz **`JuegoCartas`** para que sus métodos puedan lanzar excepciones como se muestra abajo. El método valor podrá lanzar una excepción **`CartaInvalidaException`** si recibe como parámetro una carta inválida para ese juego, y el método elegir carta podrá propagar la excepción **`CartaInvalidaException`** si se disparase al procesar la cartas de la mano o de la mesa. Completa el siguiente código para que la salida del programa sea la dada abajo. En concreto,

se pide:

(1) implementar la nueva interfaz **`JuegoCartas`** mediante una clase para el **`JuegoAbsurdo`**, en que el **valor** de *cada carta es el doble del número que forma parte de la carta (sin importar el palo de ésta), pero sólo se consideran válidas las cartas con número entre 3 y 9 (ambos inclusive)*; además en dicho juego absurdo, al **elegir una carta** de la mano se *devuelve la que tenga valor más alto entre la primera y la segunda carta de la mano* (o la primera, si solo hay una o si ambas tiene el mismo valor) sea cual sea el orden de su colocación en la mano;

(2) añadir código Java para definir y procesar excepciones, incluyendo (si fuese necesario) las partes de la clase **`Carta`** del apartado 1 que necesitasen ser modificadas), y

(3) completar el método **`main()`** para que su salida sea la que se muestra abajo.

```
package prueba3;       // primera línea deber ir en todos los archivos *.java
import java.util.*;    // segunda línea irá en todos los archivos que sea necesario

public interface JuegoCartas {
       /**  cada juego de cartas debe definir el valor de cada carta */
       public Integer valor(Carta carta) throws CartaInvalidaException;

       /**   cada juego implementa la estrategia básica para que el jugador que tiene el turno
             pueda elegir una carta de las que tiene en su mano conociendo las cartas
             que están visibles sobre la mesa jugadas anteriormente por otros jugadores */
       public Carta elegirCarta(Set<Carta> mesa, Set<Carta> mano) throws CartaInvalidaException;
}
```

```
public class Main2 {
   public static void main(String[] args)_____________________________ { //completar si fuese necesario
        JuegoCartas juego = new JuegoAbsurdo();

        Carta[] cartas = {new Carta(Palo.OROS,10,juego),  new Carta(Palo.COPAS,7,juego)};
        for (Carta c: cartas) // no hay excepción, pero las cartas invalidas dan valor cero
           System.out.println(c.valor());

        List<Carta> lista = Arrays.asList(cartas);
        HashSet<Carta> mano1 = new HashSet<Carta>(lista);
        try {
               System.out.println("carta elegida:" + juego.elegirCarta(null, mano1));
        } _____________________________ { // completar espacio subrayado
               System.out.println("excepción con mano1: " + e.getMessage());
        }
    }
}
```

Salida:

```
0
14
excepción con mano1: Se pide valor de carta no valida en este juego OROS:10
```

Prueba 3 de Evaluación Continua

Análisis y Diseño de Software (2012/2013)

Contesta en esta misma hoja

Apellidos:
Nombre:

Apartado 3 (2.5 puntos)

Se quiere reprogramar la clase **`Carta`** del apartado 1 para hacerla genérica y parametrizable mediante cualquier tipo concreto de juego de cartas. De esa forma se deberían poder declarar variables como las del método **`main()`** de abajo, que va acompañado de su salida prevista. Estos cambios no sólo afectarán a la propia cabecera de la clase sino también a algunos de sus componentes y otras clases o interfaces introducidas en el apartado 1, pero **no es necesario que reescribas todo el código anterior**.
Para este apartado se pide rellenar los espacios señalados en el código de abajo añadiendo una breve explicación para cada uno.

```
package prueba3;       // primera línea deber ir en todos los archivos *.java
import java.util.*;    // segunda línea irá en todos los archivos que sea necesario


public interface JuegoCartas {

        public Integer valor(_____________________________   carta);   // completar


        public _____________________ elegirCarta( ___________________________________ mesa,  // completar

                                                  ___________________________________ mano); // completar
} // end JuegoCartas


public class Carta_____________________________________________________   {  // completar

        private Palo palo;
        private Integer numero;

        private ________________________________ juego;

        public Carta(Palo p, Integer num, ____________________________________ j) {  // completar
             this.palo = p; this.numero = num;  this.juego = j;
       }

        public boolean equals( ____________________________________) {        // completar
           // no es necesario repetir toda la implementación del método
           // basta con indicar si hubo algún cambio respecto a la solución dada en el apartado 1



        } // end equals
} // end Carta

public class Main3 {
        public static void main(String[] args) {
            // se da por supuesto que las clases JuegoSimple y JuegoAbsurdo han sido
            // redefinidas y adaptadas correctamente a la versión genérica de la clase Carta
            JuegoSimple juegoSimple = new JuegoSimple();
            JuegoAbsurdo juegoAbsurdo = new JuegoAbsurdo();

            Carta<JuegoSimple> c1 = new Carta<JuegoSimple>(Palo.OROS,6,juegoSimple);
            Carta<JuegoAbsurdo> c2 = new Carta<JuegoAbsurdo>(Palo.OROS,6,juegoAbsurdo);
            // el valor de la carta depende del juego con el que se ha creado ver apartados 1 y 2
            System.out.println(c1 + " en su juego tiene valor: " + c1.valor());
            System.out.println(c2 + " en su juego tiene valor: " + c2.valor());
        }
}
```

Salida:

```
OROS:6 en su juego tiene valor: 6
OROS:6 en su juego tiene valor: 12
```