# *TP – Test de Unidad, Buenas Prácticas y TDD*

## Ejercicio 1 - Encriptador

La clase EncriptadorNaive se encarga de transformar cadenas de texto para que puedan ser enviadas por una red de computadoras en forma segura. La misma posee dos métodos: encriptar y desencriptar. Encriptar recibe un String y retorna otro String. La forma de encriptar es la siguiente:

Cambia el orden de las palabras. Este algoritmo recibe un numero X y divide el texto en bloques de X palabras excepto el ultimo bloque que quedará con la diferencia. Retorna una nueva frase con los bloques de palabras en orden inverso.

Por su parte, el método desencriptar realiza la inversa. Recibe un String como parámetro y retorna otro String donde realizó la desencriptación.
El protocolo de la clase debe ser el siguiente:

public String encriptar(String texto)
public String desencriptar(String texto)

Realice los test de unidad necesarios para la clase EncriptadorNaive y su método encriptar y desencriptar. ¿En que paquete debe ubicar a la clase que testea a EncriptadorNaive?

## Ejercicio 2 – TDD y Test de Unidad

Explique y desarrolle el significado de los siguientes postulados de TDD y Test de Unidad.

- Mantener en forma exhaustiva una suite de tests.
- No deben utilizarse para testear otros objetos del dominio.
- Comunicar la intención del test.

## Ejercicio 3 – Póquer

Supongamos una aplicación que da soporte a un juego de póker. Dicha aplicación cuenta con la clase **PokerStatus** que está encargada de verificar si en una ronda del juego, un jugador ha recibido, en un conjunto de 5 cartas, un juego de póquer. Un jugador obtiene póquer si de las cinco cartas recibidas, cuatro cartas iguales en su valor (ejemplo 4 ases, 4 reyes).
La clase **PokerStatus** define el método **verificar(String,String,String,String,String)** el cual recibe como parámetro la representación de las cinco cartas y retorna un booleano que indica si hubo póquer o no.
Para simplificar el problema cada carta cuenta de una representación formada por un String con dos o tres letras. Una de las letras representa el palo o figura de la carta (P = picas, C = corazones, D = diamantes, T = tréboles).
Mientras que las otras, la numeración, del 1 al 10 más J, Q y K. Por ejemplo, el dos de picas, en el sistema se puede representar con el texto “2P”; mientras que la reina de Diamantes se representa con el texto “QD” y el diez de diamantes con el string “10D”. En este ejercicio, **debe realizar el test** para el método de la clase PóquerStatus que recibe cinco cartas codificadas en forma textual y debe indicar si entre las mismas, existe

póquer o no. Asuma que la codificación recibida es correcta, es decir, una carta jamás tendrá un texto que no represente a una en un mazo.

- 1) Define diferentes escenarios para el método verificar.
- 2) Implemente el test pedido siguiendo la metodología TDD.
- 3) Identifique en su test los elementos: Setup, exercise, verify, teardown.
- 4) Programe lo necesario para pasar el test.

## Ejercicio 4 – Póquer continuación

Luego del éxito de la clase PokerStatus, los desarrolladores decidieron continuar con diferentes verificaciones en base a las reglas del juego.
Ahora se desea que además de verificar, además del póquer la existencia de Color y Trio. El color se da cuando las cinco cartas son del mismo color y palo. Mientras que el trio se da cuando tres cartas poseen el mismo valor.
La extensión ahora requiere que el método verificar retorne un String que puede ser Poquer, Color, Trio o Nada en caso que no hayan encontrado ninguna jugada.

. 1) Defina diferentes escenarios para el método verificar.
. 2) Modifique y extienda los test definidos anteriormente siguiendo TDD.
. 3) Identifique en sus tests los elementos: Setup, exercise, verify, teardown.
. 4) Programe y modifique lo necesario para pasar el test.

## Ejercicio 5 – Cartas de póquer

Los desarrolladores del póquer se dieron cuenta que la representación de las cartas usando strings es poco útil ya que ahora quieren comparar las cartas para saber cual es mayor que otra. Por eso deben implementar una clase Carta que representa a las cartas de un mazo. De una carta se puede conocer el valor y el palo. Por ejemplo, 4C ahora se representaría con una instancia de Carta. También debe ser posible saber si el valor de una carta es superior a otra, y si poseen el mismo palo.
1) Implemente los test para la clase Carta siguiendo la metodología TDD.
2) Modifique los realizado anteriormente para que el método verificar ahora reciba instancias de Carta en lugar de Strings. No abandone los legados de TDD. Tampoco los de POO.

## Ejercicio 6 - Mockito

Instalá el framework para mock objects Mockito (http://code.google.com/p/mockito/ ) y resuelva las siguientes consignas:

1. Implementar los test del ejercicio anterior (verificar con 5 Cartas) utilizando mockito.
2. ¿Cómo se indica en mockito que el objeto mock debe recibir un secuencia de mensajes particular en un orden preestablecido? Agregue un ejemplo.
3. ¿Cómo hacer para que un objeto mock este preparado para recibir algunos mensajes sin importar el orden o la obligatoriedad de recibirlos? Agregue un ejemplo.
4. Es posible anidar envío de mensajes con mockito. Si es posible, como se hace.
5. ¿Como es la forma de verificación con mockito?

## Ejercicio 7 – JUnit

Investigue y responda las siguientes consignas.

1. ¿Que es una TestSuite de Junit? Cree una para ejecutar todas las clases test del ejercicio 4.
2. ¿Como puede hacer en JUnit para que un una clase test se ejecute solamente cuando se cumple una determina condición?
3. Escriba una sentencia assertThat en la cual para una frase se valide que todas las palabras poseen mas de 5 letras y que todas las palabras incluyan el texto “test” , o por el contrario todas las palabras sean mas cortas de 5 letras y alguna de ellas comience con la letra “a”.

## Ejercicio 8 – Jugadas para el póquer

Nuevamente, es necesario agregar funcionalidad para que las jugadas puedan compararse y saber cual puede ser el ganador de una mano. El orden de importancia entre jugadas es el siguiente: Póquer le gana a Color y Color le gana a Trio. La relación es transitiva. Por su parte, si las jugadas son iguales se desempata por el valor de las cartas que la componen. El orden de las cartas es el siguiente: A, K, Q, J, 10, ..., 2. En orden de importancia descendente. Es decir, un trio con tres ases le gana a un trio con tres K.

. 1) Identifique los casos a testear.
. 2) Defina los test para el modelado de jugadas. Recuerde seguir la metodología TDD y la POO!.
. 3) Identifique en sus tests los setup, excercise, verify, teardown y doc. ¿Qué tipo de Test Doubles usó?

. 4) Actualice los test programados anteriormente para que el verificar ahora retorne instancias de jugadas.

## Ejercicio 9 – Test Doubles

. 1) ¿Qué son los test doubles?
. 2) Enumere los tipos de test doubles y de ejemplos concretos de uso. Indique como usaría, cuando es posible, mockito para emular los diferentes tipos de tipos de test doubles.