Desarrollo de Software
Prof. Carlos Alonzo
Tercer Examen Parcial
Enero 2023 **Nombre:** _____

El examen es individual. Lea con detenimiento y tranquilidad cada uno de los enunciados.

Pregunta	1	2	3	4	5	6	Total
Puntos	4	3	2	2	3	6	20
Nota							

1. (4 puntos) Enumere los cuatro (4) pilares de un buen test unitario.

2. (3 puntos) Las pruebas de caja blanca (*White-box testing*) es un método de prueba que verifica el funcionamiento interno de la aplicación. Las pruebas se derivan del código fuente, no de los requisitos ni de las especificaciones. Las pruebas resultantes de las pruebas de caja blanca a menudo son frágiles, ya que tienden a acoplarse estrechamente con la implementación específica del código bajo prueba.

   Explique por qué este método no es resistente a la refactorización.

3. (2 puntos) *"Se le transfiere la responsabilidad de crear instancias de objetos complejos y/o agregados y puede no tener responsabilidad en el modelo de dominio, pero sigue siendo parte del diseño del dominio"*. Cuál es el patrón táctico DDD que se describe ?

4. (2 puntos) Cuando se habla de pruebas unitarias (*Unit Testing*) no se hace referencia con unidad ni a un método, ni a una clase, ni a un módulo. Es decir, la unidad no representa ningún elemento estructural. Entonces, a qué se refiere el término *unidad* en las pruebas unitarias ?

5. (3 puntos) El principio SOLID de Inversión de Dependencias y la técnica de Inyección de Dependencias por Constructor favorecen que los tests unitarios sean más rápidos mediante el uso de *mocks*. Explique por qué con un ejemplo.

6. (6 puntos) ***Desarrollo de Software con Domain-driven Design y Arquitectura Hexagonal***

   Lea con detenimiento el siguiente código fuente en *TypeScript*:

Desarrollo de Software
Tercer Examen Parcial
Enero 2023

```
class Result<T> {
  resultado: T;
  error?: Error;

  isError: boolean = false;

  constructor(r: T) {
    this.resultado = r;
  }

  setResultado(r: T) { this.resultado = r; }

  setError(e: Error) { this.error = e; }
}

interface CursosRepositorio {
  getSuscriptores(cursoId: number): number[];
}

interface ISenderPush {
  send(userId: number): void;
}

interface IServicio<TComando, TResultado> {
  execute(c: TComando): Result<TResultado>;
}
```

---

1. **[2 puntos]** En el código fuente existe un *code smell* por la ausencia del uso de un patrón táctico de DDD. Indique cuál es ese patrón y dónde debe aplicarlo.
2. **[4 puntos]** Complete el código en TypeScript implementando el servicio de aplicación *CursoBloqueadoNotificarServicio*. Este servicio se encarga de enviar una notificación push a todos los suscriptores de un curso que se ha bloqueado. Este servicio de aplicación debe implementar la interface *IServicio*. En este servicio se debe hacer un buen manejo de los errores mediante la abstracción *Result*. **Usted debe implementar completamente la clase *CursoBloqueadoNotificarServicio* que corresponde al servicio de aplicación que se le pide, así como también cualquier clase adicional que no este en el código proporcionado en el enunciado.**