**Universidad de Santiago de Chile**

**Facultad de Ingeniería**

**Departamento de Ingeniería Informática**

**Paradigmas de Programación**

**Paradigmas de Programación**

**Proyecto semestral de laboratorio**

Versión 1.1 - actualizada al 27/06/2025   
- Sólo cambios de puntajes por cada RF, se arregla la numeración de los RF (había dos RF nombrados como RF10, se dejan como RF10 y RF10b para no romper el resto de numeración) y se agrega un menú de ejemplo. **No hay cambios de lógica ni RF**

Versión 1 - actualizada al 8/06/2025

# Laboratorio 3 (Paradigma Orientado a Objetos - Lenguaje Java)

(Cambios menores pueden incorporarse en futuras versiones a fin de aclarar o corregir errores)

(Sus dudas las puede expresar en este mismo enunciado, incluso puede responder a preguntas de compañeros en caso de que conozca la respuesta)

**Enunciado General: Procure consultar los aspectos generales del proyecto de laboratorio en el** [**documento general.**](https://docs.google.com/document/d/1cj6CsEN2ThY_L2Wk880z7Y03Kk1vxpoqY9LVmUQPcuQ/edit?usp=sharing)

**Fecha de Entrega:** Ver calendario clase a clase donde se señala el hito

**Objetivo del laboratorio:** Aplicar conceptos del paradigma de programación Orientado a Objetos usando el lenguaje de programación JAVA en la resolución de un problema acotado.

**Resultado esperado:** Juego CAPITALIA

**Profesor responsable: Gonzalo Martinez** (gonzalo.martinez@usach.cl) y **Edmundo Leiva** ([edmundo.leiva@usach.cl](mailto:edmundo.leiva@usach.cl)) (al hacer consultas en este documento, procurar hacer la mención a ambos profesores para que las notificaciones de sus consultas lleguen a la persona correspondiente)

**Recomendaciones:** El laboratorio está diseñado como un conjunto de ejercicios a abordar bajo cada paradigma. En este sentido, el desarrollo del laboratorio constituye un espacio para practicar y prepararse además para la evaluación de cátedra del correspondiente paradigma. Por tanto, se recomienda incorporar en sus hábitos de estudio/trabajo el desarrollo de las funcionalidades de forma diaria.

La nota comienza de 1.0 por lo que para alcanzar la nota de aprobación (4.0) usted debe sumar el puntaje de cada RF. Cada puntaje expuesto se mide en décimas, donde por ejemplo, 0.4 equivale pasar de un 1.0 a un 1.4. La nota máxima será un 7.0 aún si el puntaje se desborda. Procure destinar tiempo para analizar y hacer una propuesta de diseño para el laboratorio completo antes de proceder a la implementación de la solución. No es necesario que sus funciones implementen comprobación de tipo, esto es opcional.

## **Requerimientos No Funcionales (RNF)**

**Algunos son ineludibles/obligatorios, esto quiere decir que al no cumplir con dicho requerimiento, su proyecto será evaluado con la nota mínima.**

**Requerimientos No Funcionales. Algunos son ineludibles/obligatorios, esto quiere decir que al no cumplir con dicho requerimiento, su proyecto será evaluado con la nota mínima.**

1. **(obligatorio) Autoevaluación:** Incluir autoevaluación de cada uno de los requerimientos funcionales solicitados.
2. **(obligatorio) Lenguaje y herramientas de trabajo:** En el presente laboratorio, se debe realizar una implementación basada en **Java.**
3. La implementación de este laboratorio debe usar un SDK preciso, concretamente OpenJDK versión 11 ([link](https://adoptopenjdk.net/)). **Si bien existen versiones mayores de OpenJDK, tales como la versión 16 a 21, el laboratorio será ejecutado con la versión 11**. La subversión no es relevante (11.x.x).

**En este laboratorio no hay límite en el uso del lenguaje, puede usar cualquier librería o método provisto por el lenguaje.**

1. Puede utilizar un IDE tal como Netbeans o IntelliJ IDEA Community Edition ([link](https://www.jetbrains.com/es-es/idea/download/other.html)).
2. Para el proceso de compilación, se debe integrar a su proyecto la herramienta *Gradle* ([link a guía para crear proyecto Java usando Gradle](https://docs.google.com/document/d/1_bkG1r5-1q_-n2Ilv4GM5d19ZGgrxZ2GQjkPEVIcm8c/edit?usp=sharing)). El entregable final debe ser un proyecto creado con el IDE integrado con Gradle (y Gradle Wrapper) e incluyendo solo código fuente y archivos de configuración (no debe incluir archivos binarios ni archivos .class). **No se pueden utilizar librerías de terceros, tales como, Lombok u otros** excepto lo indique explícitamente el enunciado.

**ADVERTENCIA**: su Informe debe especificar las instrucciones de compilación, la herramienta de compilación usada (Gradle) y el ambiente de desarrollo donde se ejecutó el entregable (Sistema Operativo y versión exacta de JDK).

En caso de no poder compilar y ejecutar su proyecto con las instrucciones que usted indique, el laboratorio no será revisado. **Asimismo, si entrega su laboratorio sin Gradle, su laboratorio no será evaluado**.

En general debe solo usar bibliotecas que son parte del *core* de Java (lo nativo de Java lo puede encontrar en este [enlace](https://docs.oracle.com/en/java/javase/11/docs/api/allpackages-index.html)) , por lo que no se debe agregar dependencias externas al proyecto, con excepción de las indicadas por sus profesores que redactaron este enunciado.

**Independientemente si es una aplicación por consola o por interfaz gráfica, sólo debe realizar un entregable. Asimismo, su aplicación debe ser Java. Solo debe realizar un entregable con un lenguaje de programación.**

1. **(obligatorio) Interacciones con el programa:** Todas las interacciones con el programa deben ser mediante **consola/terminal.** Puede recurrir al uso de System.in y System.out (en Java). Para sus funcionalidades, solo se permite el uso de la biblioteca estándar de Java (sin tener dependencias externas, como las disponibles en formato JAR).
2. **(obligatorio) Uso del paradigma:** Su solución debe demostrar la acción del paradigma orientado a objetos. No basta con que su solución esté implementada en Java. Su diseño y correspondiente implementación debe seguir los lineamientos del paradigma Orientado a Objetos. **Si usted entrega todo contenido en un solo archivo Main y sin ocupar clases, se reprobara inmediatamente.**
3. **(obligatorio) Prerrequisitos:** Para cada funcionalidad se establecen prerrequisitos. Estos deben ser cumplidos para que se proceda con la evaluación de la funcionalidad implementada. Ej: Para evaluar la funcionalidad *add*, debe estar implementada la funcionalidad *authentication*.
4. **(1 pto) (obligatorio) Documentación JavaDoc:** Se debe documentar el código indicando una breve descripción de las clases e interfaces creadas, sus atributos, métodos y relaciones. Utilice comentarios como lo presenta Javadoc ([link](https://www.baeldung.com/javadoc)).

**Debe generar y entregar en el zip de entrega la documentación JavaDoc en formato HTML a partir del formato de comentarios previamente mencionado.**

1. **(1 pto) Organización del código:** Se debe cuidar la organización del código (orden y claridad). Procure que su diseño de clases no viole los principios de bajo acoplamiento y alta cohesión que muestra la adopción de parte del/la estudiante de las buenas prácticas de programación.
2. **(1.5 pto) Diagrama de análisis:** Como parte de su Informe de Laboratorio, debe incluir un diagrama de clases UML a nivel de análisis que describa las entidades y relaciones del problema abordado. Este diagrama se debe crear **ANTES** del proceso de desarrollo del software (antes de la etapa de codificación o programación).
3. **(1.5 pto) Diagrama de diseño:** Como parte de su Informe de Laboratorio, debe incluir un diagrama de clases UML **tras la implementación de la solución (es decir, DESPUÉS DE SU IMPLEMENTACIÓN)**, este diagrama debe ser coherente con la implementación en código de su solución incluyendo todas las clases de su código. **Este diagrama se debe crear después del desarrollo de la solución.**
4. **(1 pto) Uso de git:** Historial de trabajo en GitHub tomando en consideración la evolución en el desarrollo de su proyecto en distintas etapas. Se requieren **al menos 10 commits** distribuidos en un periodo de tiempo **mayor o igual a 2 semanas**. Los criterios que se consideran en la evaluación de este ítem son: fecha primer commit, fecha último commit, total commits y máximo de commits diarios. A modo de ejemplo (y solo como una referencia), si hace todos los commits el día antes de la entrega del proyecto, este ítem tendrá 0 ptos. De manera similar, si hace dos commits dos semanas antes de la entrega final y el resto los concentra en los últimos dos días, tendrá una evaluación del 25% para este ítem (0.25 pts). Por el contrario, si demuestra constancia en los commits (con aportes claros entre uno y otro) a lo largo del periodo evaluado, este ítem será evaluado con el total del puntaje.

**Requerimientos Funcionales (RF)**

**La nota correspondiente al apartado de RF comienza en 1.0 y por cada RF correcto dicho puntaje escrito en el enunciado se suma a la nota base.**

**Para que el requerimiento sea evaluado, DEBE cumplir con el prerrequisito de evaluación y requisito de implementación. En caso contrario la función no será evaluada.**

**Requerimientos Funcionales. Para que el requerimiento sea evaluado, DEBE cumplir con el prerrequisito de evaluación y requisito de implementación. En caso contrario la función no será evaluada.**

1. **RF01 Clases y estructuras que forman el programa (1 pts)**.

Especificar e implementar abstracciones apropiadas para el problema. **Recomendamos leer el enunciado completo y ver el material complementario presentado al comienzo de este enunciado a fin de que analice el problema y determine el o los TDAs y representaciones apropiadas para la implementación de cada uno**. Luego, planifique bien su enfoque de solución de manera que los TDAs y representaciones escogidos sean aplicables a ambos tipos de funciones.

Para la implementación debe regirse por la estructura de especificación e implementación de TDA vista en clases: Representación, Constructores, Funciones/Métodos de Pertenencia, Selectores, Modificadores y Otros Métodos. Procurar hacer un uso adecuado de esta estructura a fin de no afectar la eficiencia de sus Métodos. En el resto de los Métodos se debe hacer un uso adecuado de la implementación del TDA (ej: usar selectores, modificadores, constructores, según sea el caso. No basta con implementar un TDA y luego NO hacer uso del mismo). **Solo implementar las clases y métodos estrictamente necesarios dentro de esta estructura**.

**A excepción de la clase Main, el nombre del resto de clases debe contener su nombre y rut (Clase\_NombreApellido\_Rut.java).**

**Si alguna clase contiene métodos que no son utilizados se procede a descontar puntaje relativo a este RF.**

Como parte del diseño orientado a objetos de su solución, **considere como mínimo modelar las siguientes entidades (y sus respectivas relaciones)** dentro de su programa:

* 1. **Main**
  2. **Game: Juego (Game\_NombreApellido\_21xxx.java)**
  3. **Board: Tablero (Board\_NombreApellido\_21xxx.java)**
  4. **Player: Jugador (Player\_NombreApellido\_21xxx.java)**
  5. **Property: Propiedad (Property\_NombreApellido\_21xxx.java) (También se debe incluir el TDA Hotel en otra clase archivo java)**
  6. **Card: Carta (Card\_NombreApellido\_21xxx.java) (También se debe incluir el TDA Comunidad y CartaSuerte cada uno en otra clase archivo java)**

| **Prerrequisitos para evaluación** | Ninguno |
| --- | --- |
| **Requisitos de implementación** | - Como ***mínimo*** su diseño debe contemplar los TDA previamente descritos**. Usted puede crear más clases y TDAs si lo considera necesario. El diseño de software e implementación del programa queda a su disposición. No evaluamos el diseño de software.**  - Especificar representación de manera clara para cada clase implementada (en el informe y en el código a través de comentarios en formato JavaDoc). Luego implementar **constructores, getters, setters, modificadores** y otros métodos según lo requerido en los requisitos a continuación.  - Sólo debe implementar los getters + setters y métodos en general que efectivamente se utilizan en su código. **No debe existir código declarado, pero que no se utilice en el resto de las clases declaradas**. |

1. **RF02. (1 pts) Menú interactivo por terminal\*:** su programa debe incluir un menú por terminal/consola que permita la interacción del usuario con la solución, implementando las entidades y funcionalidades que permitan interactuar con las clases *especificadas en el RF1. Si la funcionalidad del RF no es expuesto mediante el menú interactivo, no será revisado.* El sistema debe proveer retroalimentación al usuario de las acciones realizadas por él (si la acción pudo concretarse con éxito o no).

Las instrucciones de funcionamiento de su solución deben estar explicadas y documentadas en su Informe. **IMPORTANTE**: La implementación de este menú debe manejarse de manera independiente del resto de las clases que conforman la solución al problema establecido.

* **IMPORTANTE: EN ESTA ENTREGA NO EXISTE UN SCRIPT DE EJECUCIÓN. TODAS LAS INTERACCIONES SE REALIZARÁN MEDIANTE ESTE MENÚ. AL FINAL DEL DOCUMENTO SE ENCUENTRA UN EJEMPLO DE MENÚ. EL DISEÑO E IMPLEMENTACIÓN DEL MENÚ QUEDA A DISPOSICIÓN DEL ESTUDIANTE.**

**IMPORTANTE: A diferencia de otros laboratorios, dada la naturaleza del paradigma el dominio y recorrido queda a su criterio**

### **3. RF03. (0.1 pts) TDA Jugador (Player) - constructor**. Método que permite crear un jugador.

| **Prerrequisitos para evaluación (req. funcionales)** | 1-2 |
| --- | --- |
| **Descripción** | Un Jugador contiene como mínimo la información de:  id (int) X nombre (String) X dinero (int) X propiedades (list Propiedad) X posicionActual (int) X estaEnCarcel (boolean) X totalCartasSalirCarcel (int) |

### **4. RF04. (0.1 pts) TDA Propiedad (Property) - constructor**. Método que permite crear una propiedad en el juego

| **Prerrequisitos para evaluación (req. funcionales)** | 3 |
| --- | --- |
| **Descripción** | Una propiedad contiene como mínimo la información de:  id (int) X nombre (string) X precio (int) X renta (int) X dueño (Player) X casas (**queda a criterio del estudiante su representación**) X estaHipotecada (**queda a criterio del estudiante su representación**)  Relativo a hotel verificar siguiente RF |

### **5. RF05. (0.1 pts) TDA Hotel - constructor**. Método para crear una Propiedad Hotel

| **Prerrequisitos para evaluación (req. funcionales)** | 4 |
| --- | --- |
| **Requisitos de implementación** | Hotel es un tipo de propiedad.  Debe ser implementado mediante uso de interface o alguna forma de herencia.  La justificación de la técnica seleccionada debe estar contenida en el informe capítulo Diseño de la Solución. |

### **6. RF06. (0.1 pts) TDA Carta - constructor**. Crear una carta en el juego

| **Prerrequisitos para evaluación (req. funcionales)** | 5 |
| --- | --- |
| **Requisito de implementación** | Usted debe diseñar e implementar sus propios métodos y/o clases que den acción a las cartas  El TDA Carta contiene como mínimo:  id (int) X descripcion (String)  accion (métodos y/o clases) |

### **7. RF07. (0.1 pts) TDA CartaComunidad - constructor**. Crear una carta comunidad

| **Prerrequisitos para evaluación (req. funcionales)** | 6 |
| --- | --- |
| **Requisitos de implementación** | CartaComunidad es un tipo de Carta.  Debe ser implementado mediante uso de interface o alguna forma de herencia.  La justificación de la técnica seleccionada debe estar contenida en el informe capítulo Diseño de la Solución. |

8. **RF08. (0.2 pts) TDA CartaSuerte - constructor.** Crear una carta suerte

| **Prerrequisitos para evaluación (req. funcionales)** | 7 |
| --- | --- |
| **Requisitos de implementación** | CartaSuerte es un tipo de Carta.  Debe ser implementado mediante uso de interface o alguna forma de herencia.  La justificación de la técnica seleccionada debe estar contenida en el informe capítulo Diseño de la Solución. |

### **9. RF09 (0.1 pts) TDA Tablero - constructor**. Crear un tablero de CAPITALIA

| **Prerrequisitos para evaluación (req. funcionales)** | 8 |
| --- | --- |
| **Requisitos de implementación** | Observación: Las casillas especiales incluyen la casilla de salida (posición 0), cárcel (usted define la posición), casilla de comunidad y suerte.  Aclaración: salida siempre es posición 0 y cárcel usted lo define pero debe ser constante durante la ejecución de su entrega.  Un tablero contiene como mínimo:  propiedades (lista Property) X cartas (list Carta) X casillas-especiales (**usted define el tipo de dato y su representación**) |

### 

### 

### **10. RF10. (0.3 pts) TDA Juego - constructor**. Método que crea una partida de CAPITALIA

| **Prerrequisitos para evaluación (req. funcionales)** | 9 |
| --- | --- |
| **Requisitos de implementación** | **Se mantienen las reglas y aclaraciones del laboratorio 2.**  El TDA Juego debe contener como mínimo la información de:   * Lista de jugadores * dineroBanco (int): La cantidad total de dinero disponible en el banco. En el ejemplo es 20000. * numeroDados (int): Cantidad de dados que se usarán en el juego. Según las reglas, puede ser entre 1 y 4 dados. Cómo mínimo usted debe soportar en su juego 2 dados. * turnoActual (queda a su criterio cómo implementar este dato) * tasaImpuesto (int): El porcentaje de impuesto que pagarán los jugadores por sus propiedades al completar una vuelta al tablero. * maximoCasas (int): Número máximo de casas que se pueden construir en una propiedad antes de poder construir un hotel. * maximoHoteles (int): Número máximo de hoteles que se pueden construir en una propiedad.   Aclaración: Para este laboratorio el maximoCasas y maximoHoteles es para todas las propiedades sin excepciones.  Queda a disposición del estudiante cómo implementar la información mínima solicitada (clases, atributos, métodos, técnicas de polimorfismo, composición, etc) |

### **10. RF10b. (0.3 pts) TDA Juego - imprimir**. Método que imprime en pantalla el estado actual del juego con toda su información asociada

### 

| **Prerrequisitos para evaluación (req. funcionales)** | 10 |
| --- | --- |
| **Requisitos de implementación** | **Imprime el contenido del juego, esto incluye información de tablero, jugadores, propiedades.**  **El formato de cómo se imprime queda a disposición del estudiante**  **Puede usar sobreescritura (override) del método toString de la clase.** |

**Las siguientes funcionalidades no tienen un requerimiento obligatorio de asociar directamente a un TDA o Clase y/o método específico. Tampoco tienen un dominio o recorrido obligatorio. La razón de esto es que a diferencia de laboratorios pasados, acá por la naturaleza del paradigma, usted usando las técnicas de Paradigma Orientado a Objetos debe diseñar e implementar su sistema a través de la implementación de los métodos, clases y/o atributos que crea pertinentes para dar resolución a cada uno de los siguientes RF.**

### **11. RF11. (0.1 pts) Cargar datos iniciales**. Al momento de iniciar el juego ya debe existir un tablero cargado con propiedades, cartas suerte y comunidad.

| **Prerrequisitos para evaluación (req. funcionales)** | 10 |
| --- | --- |
| **Requisitos de implementación** | Esta funcionalidad se ejecuta al momento de ejecutar un nuevo juego (jugar nueva partida). Esta funcionalidad cargará los siguientes datos a su sistema:  * **15 propiedades.** * **10 cartas suerte** * **10 cartas comunidad**   **Este laboratorio no tiene un script de ejecución pero si al momento de la evaluación necesitamos que ya exista un tablero creado.  Usted debe crear los objetos necesarios y cargar la información al sistema. Nosotros cómo equipo evaluador no crearemos entidades adicionales en su laboratorio ni modificaremos código así que usted debe preocupar de cargar todo lo necesario** |

### **12. RF12. (0.1 pts) Agregar Propiedad**. Agregar propiedades al tablero.

| **Prerrequisitos para evaluación (req. funcionales)** | 11 |
| --- | --- |
| **Requisitos de implementación** | Usted decide si se puede agregar 1 o N propiedades al mismo tiempo. Queda a disposición del estudiante cómo implementar este requisito (si es 1 o N). |

### **13. RF13. (0.1 pts) Agregar Jugador**. Agregar un jugador a la partida.

| **Prerrequisitos para evaluación (req. funcionales)** | 12 |
| --- | --- |
| **Requisitos de implementación** | El comportamiento es agregar un jugador a la partida y le asigna el capital inicial (Debe disminuirse del banco).  El capital inicial es de 1500 para todos los jugadores |

### **14. RF14. (0.1 pts) Obtener Jugador Actual**. Obtener el jugador cuyo turno se encuentra en curso (jugador actual)

| **Prerrequisitos para evaluación (req. funcionales)** | 13 |
| --- | --- |

### **15. RF15. (0.1 pts) Lanzar dados**. Simular el lanzamiento de N dados

| **Prerrequisitos para evaluación (req. funcionales)** | 14 |
| --- | --- |
| **Requisitos de implementación** | Cómo mínimo para efectos de un juego puede asumir que sólo son 2 dados.  Puede usar librerías internas de java para esto, por ejemplo java.util.Random  [Random (Java Platform SE 8 )](https://docs.oracle.com/javase/8/docs/api/java/util/Random.html) |

### **16. RF16. (0.1 pts) Mover Jugador**. Mover al jugador en el tablero

| **Prerrequisitos para evaluación (req. funcionales)** | 14 |
| --- | --- |
| **Requisitos de implementación** | El tablero se debe ver actualizado con el movimiento del jugador.  Se mantiene aclaración de laboratorio 1 y 2:  Este RF sólo actualiza la posición del jugador. Las reglas del juego se aplican en otro RF. |

### **17. RF17. (0.1 pts) Comprar Propiedad**. Permite comprar una propiedad

| **Prerrequisitos para evaluación (req. funcionales)** | 15 |
| --- | --- |
| **Requisitos de implementación** | Se debe verificar si el jugador tiene suficiente dinero.  El diseño e implementación de esta funcionalidad queda a su disposición. |

### **18. RF18. (0.1 pts) Calcular Renta Propiedad**. Calcular la renta de una propiedad sumando todas las casas y hoteles que tenga.

| **Prerrequisitos para evaluación (req. funcionales)** | 16 |
| --- | --- |
| **Requisitos de implementación** | **Se mantiene las reglas y aclaraciones de laboratorio 2:**  Calcular el monto de la renta que debe pagar un jugador al caer en una propiedad específica.  Para este laboratorio el cálculo de renta es:   * Renta base = valor de la propiedad (si es que no hay construcciones). * Por cada casa construida se aumenta 20% el valor de la propiedad. * Aumento adicional si tiene hotel. La renta con hotel es el doble de la renta con el máximo de casas. Ejemplo: si la renta con el máximo de casas es 100, la renta con hotel es el doble y queda en 200. * Si la propiedad está hipotecada, la renta es 0.   Si el jugador ha construido 2 casas, la renta siempre e s$14 Documento de reglas:  [2025\_01 Laboratorio (General)](https://docs.google.com/document/d/1cj6CsEN2ThY_L2Wk880z7Y03Kk1vxpoqY9LVmUQPcuQ/edit?tab=t.0#heading=h.nnf213ilkre1) |

### **19. RF19. (0.1 pts) Calcular Renta Jugador**. Calcular la renta de las propiedades de un jugador.

| **Prerrequisitos para evaluación (req. funcionales)** | 17 |
| --- | --- |
| **Requisitos de implementación** | Calcular la renta según el número de casas y hoteles.  Se mantiene reglas y aclaraciones de laboratorio 2:  Incrementar el número de casas en una propiedad, verificando que no supere el máximo permitido según el juego.  Se debe incrementar en 1 el número de casas en una propiedad, siempre y cuando no se supere el máximo de casas permitido (definido en el juego). Si ya se alcanzó el máximo, el predicado retorna la propiedad sin cambios.  Aclaración: en este laboratorio el costo de construir una casa es el valor de la propiedad definido en el TDA propiedad. |

### **20. RF20. (0.2 pts) Construir Hotel**. Construir Hotel en una Propiedad

| **Prerrequisitos para evaluación (req. funcionales)** | 18 |
| --- | --- |
| **Requisitos de implementación** | Se mantienen las mismas reglas y aclaraciones del laboratorio 2.   * Convertir las casas de una propiedad en un hotel si se tiene el máximo de casas permitido. * Este predicado convierte las casas en un hotel si la propiedad tiene el número máximo de casas (definido en el juego). Al hacerlo, establece el campo esHotel a true y el número de casas a 0. Si la propiedad no tiene el máximo de casas, el predicado unifica la propiedad sin cambios. * Aclaración: el costo monetario de un hotel se simplificó, construir un hotel es reemplazar el número máximo de casas con el hotel, no tiene un costo monetario asociado (en términos de juego el costo es que se perdieron las casas construidas) |

### **21. RF21. (0.2 pts) Pagar Renta**. Jugador paga renta a otro.

| **Prerrequisitos para evaluación (req. funcionales)** | 18 |
| --- | --- |
| **Requisitos de implementación** | Se mantienen las mismas reglas y aclaraciones del laboratorio 2.   * Esta funcionalidad transfiere dinero de un jugador a otro. * Aclaración: un jugador paga renta a otro cuando al momento de ejecutar el turno este cae en propiedad ajena. Esta funcionalidad no contempla el cálculo de renta si no que acá sólo se realiza el pago. |

### **22. RF22. (0.2 pts) Hipotecar Propiedad**. Hipotecar una propiedad

| **Prerrequisitos para evaluación (req. funcionales)** | 18 |
| --- | --- |
| **Requisitos de implementación** | Se mantienen las mismas reglas y aclaraciones del laboratorio 2.   * Recordar que dada la aclaración del comentario de 29 de mayo, hipotecar es lo mismo que su "renta", hipotecar viene siendo como un préstamo. |

### **23. RF23. (0.2 pts) Extraer Carta**. Extraer una carta del mazo.

| **Prerrequisitos para evaluación (req. funcionales)** | 18 |
| --- | --- |
| **Requisitos de implementación** | Extraer una carta aleatoria del mazo.  Queda a su disposición el diseño e implementación de esta funcionalidad. |

### **24. RF24. (0.1 pts) Verificar bancarrota**. Verificar si un jugador se encuentra en bancarrota (sin dinero).

| **Prerrequisitos para evaluación (req. funcionales)** | 18 |
| --- | --- |
| **Requisitos de implementación** | Esta funcionalidad les ayudará a terminar el juego. El juego termina cuando un jugador se encuentra actualmente sin dinero. |

### **25. RF25. (1 pts) Jugar Turno**. Ejecuta un turno completo aplicando todas las reglas del juego

| **Prerrequisitos para evaluación (req. funcionales)** | 18 |
| --- | --- |
| **Requisitos de implementación** | Ejecuta el turno correspondiente con las reglas descritas en el enunciado general.  Reglas:  [2025\_01 Laboratorio (General)](https://docs.google.com/document/d/1cj6CsEN2ThY_L2Wk880z7Y03Kk1vxpoqY9LVmUQPcuQ/edit?tab=t.0)  Queda a su disposición el diseño e implementación de esta funcionalidad. |

### **26. OPCIONAL: RF26. (2 pts) Integración con IA**. En vez de tener dos jugadores humanos que puedan interactuar con su sistema, sólo se tiene 1 jugador humano (el jugador) y el otro es Gemini IA

| **Prerrequisitos para evaluación (req. funcionales)** | RF14 |
| --- | --- |
| **Requisitos de implementación** | Integrar IA como Jugador 2. Ver instrucciones de Enunciado General para lograr la integración  Reglas:  [2025\_01 Laboratorio (General)](https://docs.google.com/document/d/1cj6CsEN2ThY_L2Wk880z7Y03Kk1vxpoqY9LVmUQPcuQ/edit?tab=t.0)  Queda a su disposición el diseño e implementación de esta funcionalidad. |

### **27. OPCIONAL: RF27. (1 pts) Exportar partida actual a un archivo**. Funcionalidad expuesta en el menú que permite exportar la partida en curso a un archivo.

| **Prerrequisitos para evaluación (req. funcionales)** | RF14 |
| --- | --- |
| **Requisitos de implementación** | El archivo y formato resultante queda a su criterio. Puede ser json/xml/csv/yaml, etc, queda a su criterio.  Para ayudar su búsqueda para la implementación de esta funcionalidad, busque por "Parser". Ejemplo: https://docs.oracle.com/cd/E21764\_01/appdev.1111/b28394/adx\_j\_parser.htm  Queda a su disposición el diseño e implementación de esta funcionalidad. |

### **28. OPCIONAL: RF28. (1 pts) Importar partida desde un archivo**. Funcionalidad expuesta en el menú que permite importar un archivo ya creado y cargar el juego actual al sistema.

| **Prerrequisitos para evaluación (req. funcionales)** | RF14 |
| --- | --- |
| **Requisitos de implementación** | El archivo y formato queda a su criterio.  Puede ser json/xml/csv/yaml, etc, queda a su criterio.  Para ayudar su búsqueda para la implementación de esta funcionalidad, busque por "Parser". Ejemplo: https://docs.oracle.com/cd/E21764\_01/appdev.1111/b28394/adx\_j\_parser.htm  Queda a su disposición el diseño e implementación de esta funcionalidad. |

**EJEMPLO DE MENÚ Y FLUJO DE EJECUCIÓN**

## ESTO ES SÓLO UN EJEMPLO, EL DESARROLLO DEL MENÚ QUEDA A LIBRE DISPOSICIÓN

### CAPITALIA - Menú Principal ###

Bienvenido al juego CAPITALIA

Seleccione una opción:

1. Jugar

2. Visualizar estado actual del tablero

3. Salir del juego

Ingrese su opción: 1

### Crear Nuevo Juego ###

--- Configuración Jugador 1 ---

Ingrese nombre del jugador 1: John Lennon

--- Configuración Jugador 2 ---

Ingrese nombre del jugador 2: Yoko Ono

--- Configuración del Juego ---

Ingrese cantidad de dados por jugador: 2

¡Juego creado exitosamente!

Comienza el juego…

### Realizar Jugada ###

Turno de: John

Lanzar dados. Dado 1: 3, Dado 2: 1. La suma da 4

John se mueve 4 lugares

John cayo en la propiedad A cuyo costo es $40.000  
Desea comprar propiedad? (S/N): S

Turno de: Yoko

Lanzar dados. Dado 1: 1, Dado 2: 1. La suma da 2

Yoko se mueve 2 lugares

Yoko cayo en la propiedad B cuyo costo es $500  
Desea comprar propiedad? (S/N): S

¿Desea jugar otra partida? (S/N): N

### Conecta4 - Menú Principal ###

1. Crear nuevo juego

2. Visualizar estado actual

3. Realizar jugada

4. Ver estadísticas generales

5. Salir del juego

Ingrese su opción: 5

¿Está seguro que desea salir? (S/N): S

¡Gracias por jugar CAPITALIA!

**Leer si tienen problemas de compilación y ejecución con Gradle**

La entrega se ejecutará vía terminal/consola usando gradle. Para esto los revisores ocupan los siguientes comandos. Deben verificar si su entrega funciona con los siguientes comandos que permiten compilar y ejecutar. No se revisará con el IDE ni tampoco se modificará **Linux/Unix:**

1. Compilación:

./gradlew build

1. Ejecución:

./gradlew run

**Windows**:

1. Compilación:

gradlew.bat build

1. Ejecución:

gradlew.bat run

Si la clase Scanner de Java, o el programa en sí con el uso del menú interactivo arroja un error, posiblemente es porque falta una configuración en el archivo de configuración de Gradle (build.gradle). Para esto usen de referencia la siguiente configuración:

**build.gradle** en Groovy (Si realizaron el archivo de configuración build.gradle en otro lenguaje, tal como Kotlin, debe buscar el equivalente de la siguiente configuración).

# IMPORTANTE: SI SU PROGRAMA FALLA EN TERMINAL/CONSOLA NECESITA AGREGAR ESTO  
plugins {

id 'application'

}

repositories {

mavenCentral()

}

# IMPORTANTE: SI SU PROGRAMA FALLA EN TERMINAL/CONSOLA NECESITA AGREGAR ESTO

run {

standardInput = System.in

}

# Acá debe ir la ruta en donde se encuentra el main, org es la carpeta de mayor jerarquía, luego la carpeta example y luego la clase Main. OJO: NO ES QUE SU PROGRAMA TENGA QUE TENER LA CARPETA org, O LA CARPETA example, ES SOLO UN EJEMPLO.

application {

mainClass = 'org.example.Main'

}

**Al momento de usar de referencia la configuración anterior, eliminen los comentarios (#)**

// mvn:

mvn run

// gradle

gradle run