

Orientación a Objetos I

2025

Explicación de práctica #9
correspondiente a los ejercicios de la
semana del 3 de Noviembre



FACULTAD DE INFORMATICA



UNIVERSIDAD
NACIONAL
DE LA PLATA

Actividades de la semana anterior

- Ejercicio 22. Sistema de pedidos
- Ejercicio 23. Poolcar
- Ejercicio 24. GreenHOOme
- Ejercicio 25. Bag

Esta semana:

- Ejercicio 26. Estadísticas del Cliente de Correo
- Ejercicio 27. Method Lookup
- Ejercicio 28. TipadOOs
- Ejercicio 29. Plataforma de Streaming



FACULTAD DE INFORMATICA



UNIVERSIDAD
NACIONAL
DE LA PLATA

Ejercicio 26 - Estadísticas del Cliente de Correo

Extienda el Ejercicio 15: Cliente de Correo:

Nos piden implementar la siguiente funcionalidad:

- cantidad de emails que tiene una carpeta
- cantidad total de emails en el cliente de correo: considerando todas las carpetas existentes.
- cantidad de mails por categoría: para cada carpeta se debe calcular y retornar en un solo objeto, la cantidad de emails categorizados por tamaño siguiendo el siguiente criterio
 - Pequeño: el email tiene un tamaño entre 0 y 300
 - Mediano: el email tiene un tamaño entre 301 y 500
 - Grande: el email tiene un tamaño mayor a 501



FACULTAD DE INFORMATICA



UNIVERSIDAD
NACIONAL
DE LA PLATA

Ejercicio 26 - Estadísticas del Cliente de Correo

Cantidad de mails por categoría: para cada carpeta se debe calcular y **retornar en un solo objeto**, la cantidad de emails categorizados por tamaño siguiendo el siguiente criterio:

- Pequeño: el email tiene un tamaño entre 0 y 300
- Mediano: el email tiene un tamaño entre 301 y 500
- Grande: el email tiene un tamaño mayor a 501

"Pequeño"	2
"Mediano"	3
"Grande"	4

Es decir, el cliente de correo, en respuesta a un mensaje por ejemplo

"**mailsCategorizados**" debería devolver un objeto que internamente tenga esta información estructurada como muestra la tabla

¿Existe algún objeto que hayamos programado o usado en un ejercicio anterior que tenga esta estructura y comportamiento?



FACULTAD DE INFORMATICA



UNIVERSIDAD
NACIONAL
DE LA PLATA

Ejercicio 26 - Estadísticas del Cliente de Correo

```
public interface Bag<T> extends Collection<T> {  
    @Override  
    boolean add(T element);  
    int occurrencesOf(T element);  
    void removeOccurrence(T element);  
    void removeAll(T element);  
    @Override  
    int size();  
}
```



FACULTAD DE INFORMATICA



UNIVERSIDAD
NACIONAL
DE LA PLATA

Ejercicio 27 - Method Lookup

Dado el siguiente modelo, y los fragmentos de códigos mostrados, marque la respuesta para cada una de las preguntas enunciadas. Sólo una opción es la correcta.

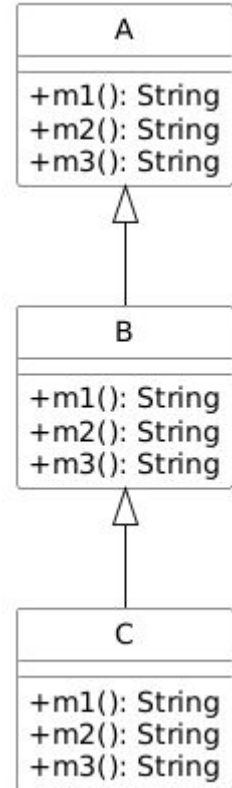
```
C c = new C();  
c.m1();
```

1. C.m1 -> B.m1 -> C.m2
2. C.m1 -> A.m1 -> A.m2
3. C.m1 -> B.m1 -> B.m2 -> A.m2
4. C.m1 -> A.m1 -> C.m2

```
C c = new C();
```

1. A.m2
2. B.m2 -> A.m2
3. C.m2
4. B.m2 -> C.m2

```
c.m2();
```

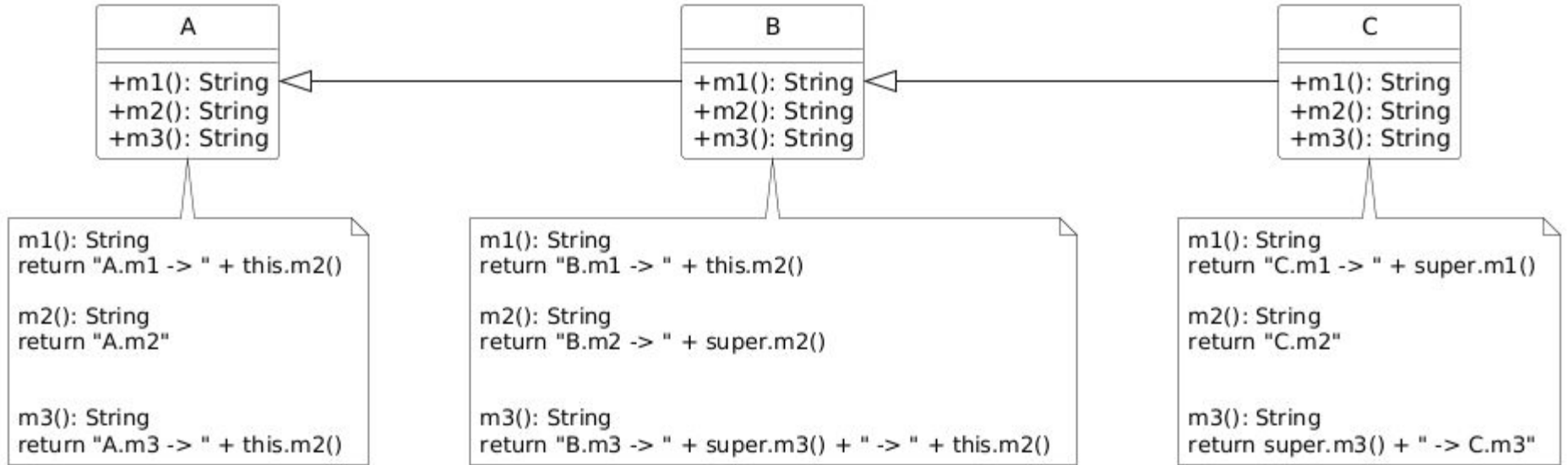


FACULTAD DE INFORMATICA



UNIVERSIDAD
NACIONAL
DE LA PLATA

Ejercicio 27 - Method Lookup



FACULTAD DE INFORMATICA



UNIVERSIDAD
NACIONAL
DE LA PLATA

Ejercicio 27 - Method Lookup

- El ejercicio nos propone un modelo UML y el código de los métodos para que podamos analizar y responder las preguntas
- Para cada pregunta, **sólo una respuesta es la correcta**
- El comportamiento no es importante, los métodos devuelven strings que se van concatenando
- Lo importante es seguir el **method lookup** correctamente para poder llegar al resultado esperado
 - Prestar atención a la jerarquía, y al uso de **this** y **super**
- Recomendamos no hacerlo con Java, sino pensar y seguir el código. Luego es posible validarlo mediante ejecución, pero es importante ser capaces de seguir el código



FACULTAD DE INFORMATICA

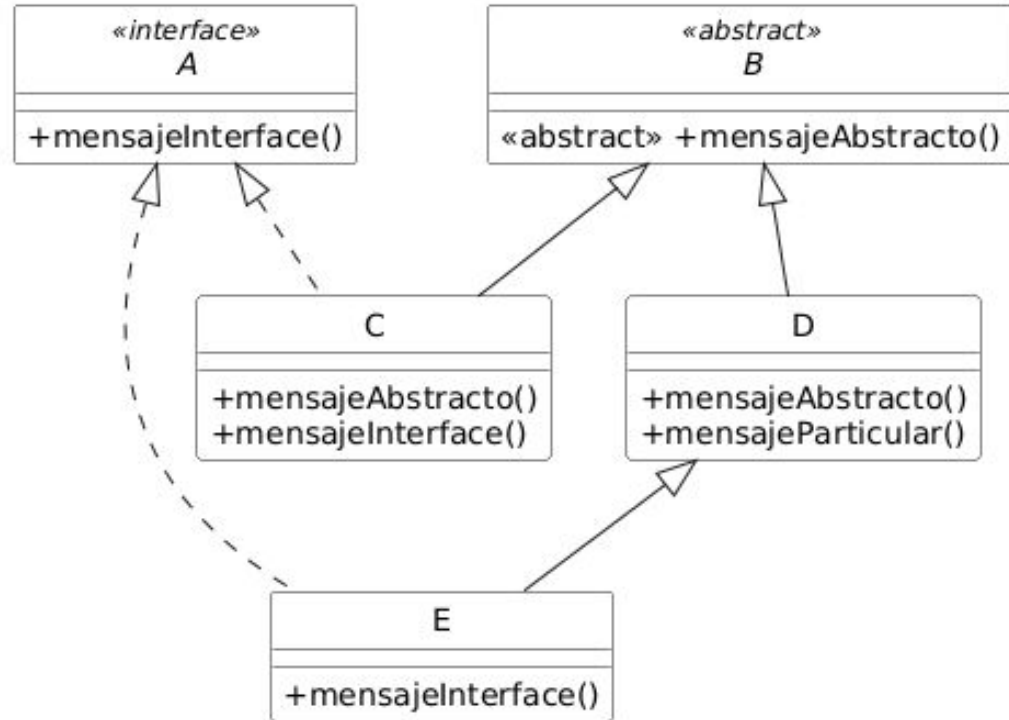


UNIVERSIDAD
NACIONAL
DE LA PLATA

Ejercicio 28 - TipadOOs

Dado el diagrama de clases UML proporcionado, complete todos los bloques de código reemplazando los signos de interrogación (???) con los tipos y métodos correctos.

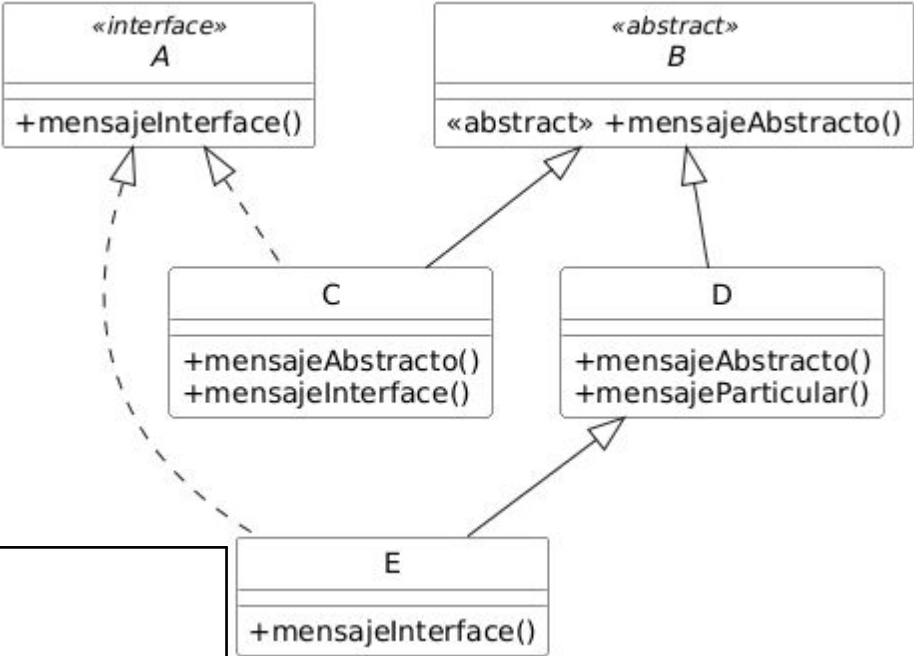
Escriba todas las combinaciones válidas posibles.



FACULTAD DE

Dado el diagrama de clases UML proporcionado, complete todos los bloques de código reemplazando los signos de interrogación (???) con los tipos y métodos correctos.

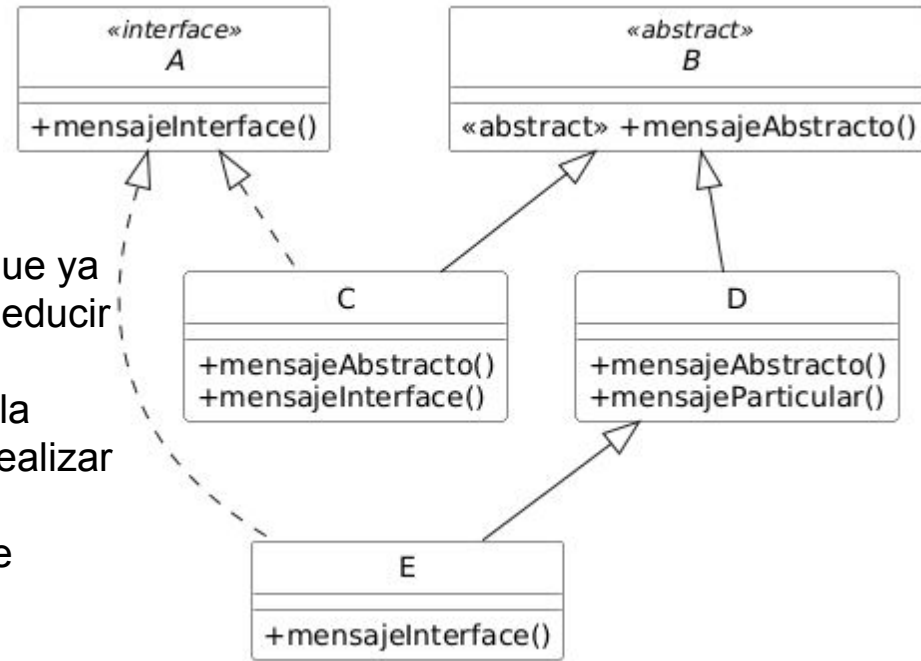
Escriba todas las combinaciones válidas posibles.

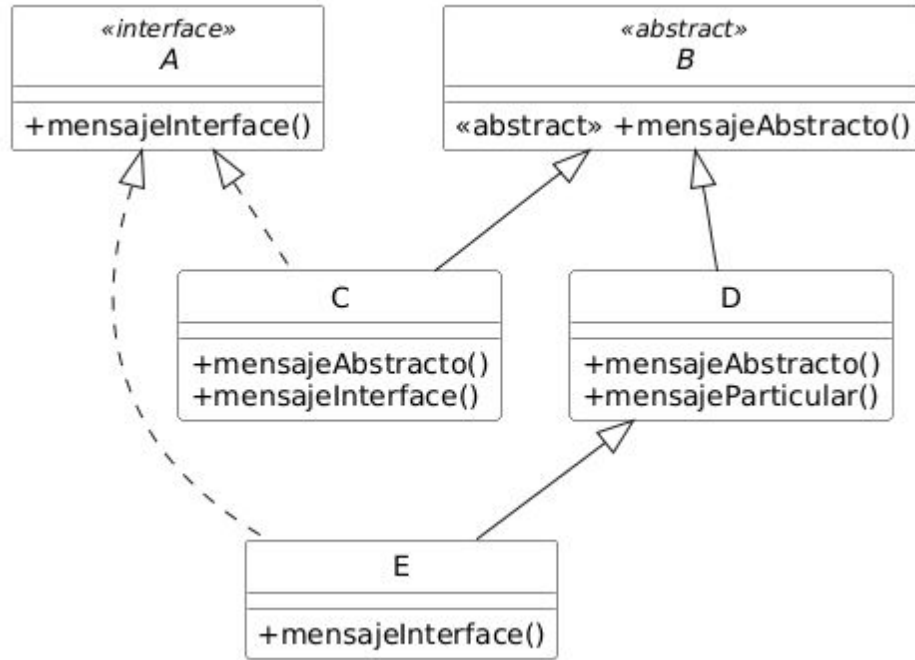


1) A objeto = new ??? (); objeto.??? ();	4) C objeto = new C (); objeto.??? ();
2) B objeto = new ??? (); objeto.??? ();	5) ??? objeto = new C (); objeto.mensajeAbstracto ();
3) D objeto = new ??? (); objeto.??? ();	6) ??? objeto = new C (); objeto.mensajeInterface ();

```
A objeto = new ??? ();  
objeto.??? ();
```

- El fragmento de código nos presenta una parte que ya está fija y una parte que nosotros tenemos que deducir a partir del diagrama
- Las clases del diagrama no tienen el código con la implementación. No lo necesitamos para poder realizar el ejercicio
- Prestar atención a los **tipos**, ya que tenemos que detectar que cosas puedo crear y asignar, y qué conjunto de mensajes soporta
- Prestar atención a interfaces y clases abstractas. Recordar que **no se pueden instanciar**, pero sí puedo definir una variable en donde su tipo es la interface o la clase abstracta
- Importancia de tipos en compilación y en ejecución (binding dinámico)
- Repasar la teoría de la semana 15 de Septiembre

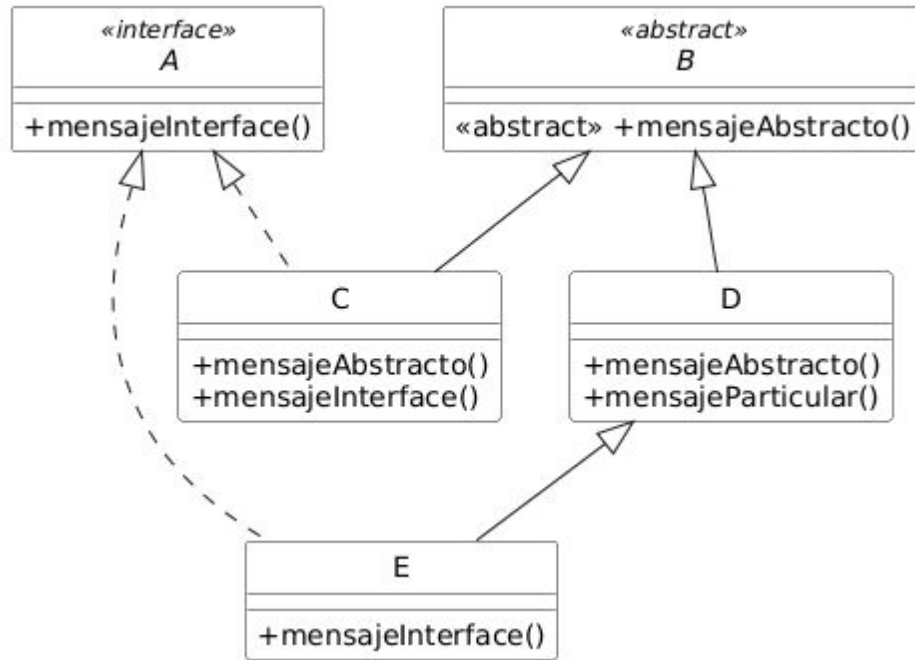




```
A objeto = new ??? ();  
objeto.??? ();
```

- ¿Que podemos crear y asignar a la variable objeto?
- ¿Qué protocolo tiene la variable objeto?
 - ¿Depende de que hayamos creado en la línea anterior?
 - ¿Se define en tiempo de compilación o en tiempo de ejecución?

Recordar: **hay múltiples combinaciones válidas** y hay que escribirlas todas



```
??? objeto = new C();  
objeto.mensajeAbstracto();
```

- Está creando y asignando un objeto de clase C, ¿con qué tipos podemos definir a la variable **objeto**?
- Además, en la línea siguiente se está enviando el mensaje “**mensajeAbstracto**”. ¿Que tipos del diagrama soportan ese mensaje en su protocolo?

Recordar: **hay múltiples combinaciones válidas** y hay que escribirlas todas

Ejercicio 29 - Plataforma de Streaming

(Extracto del enunciado, hay más funcionalidades)

Nos piden implementar la siguiente funcionalidad:

Calcular el precio base de un plan: Para los planes individuales, el precio base es \$20 por la cantidad de minutos contratados. Para los planes grupales, el precio base es de \$800 por la cantidad límite de IPs contratadas.

Monto total a cobrar: dado un cliente y un intervalo de tiempo (fecha inicial - fecha final), retorna el monto total a cobrar para ese intervalo. El monto total a cobrar es la suma del precio base según el plan, el monto por las actividades registradas y el monto por penalización.

(...sigue en el cuadernillo)



FACULTAD DE INFORMATICA



UNIVERSIDAD
NACIONAL
DE LA PLATA

Ejercicio 29 - Plataforma de Streaming

Es un ejercicio integrador, abarca todos los temas de la materia

(Extracto del enunciado)

Una plataforma de streaming registra las actividades realizadas por los clientes para calcular los montos a cobrar a cada uno. De cada cliente se conoce el nombre y su fecha de alta. Los clientes pueden contratar un plan que puede ser individual o grupal. Al momento de contratar el plan, se establece una cantidad límite en las direcciones IP desde las cuales se permite acceder a las actividades, sin generar un cargo adicional. En el caso de los planes individuales se permite acceder desde una única dirección IP; mientras que en los planes grupales se permite acceder desde varias direcciones IP, con una cantidad máxima establecida al contratar el plan. En los planes individuales se registra además la cantidad de minutos contratados (... sigue en el cuadernillo)



FACULTAD DE INFORMATICA



UNIVERSIDAD
NACIONAL
DE LA PLATA

1) **Modelado de dominio**

- i) Realice un diagrama de clases UML donde identifique las clases del dominio, atributos, y las relaciones con sus roles y cardinalidades cuando corresponda. No debe incluir comportamiento.

2) **Modelado e Implementación**

- i) Continúe completando el diagrama anterior con el comportamiento de cada clase.
- ii) Realice el Diagrama de clases UML a partir del modelo de dominio realizado en el punto anterior.
- iii) Implemente en Java la funcionalidad requerida. Puede usar las clases implementadas en los ejercicios de la práctica: DateLapse, FilteredSet, Bag, según se requiera.

3) **Pruebas automatizadas**

- i) Diseño de los casos de prueba: Enfocándose en la funcionalidad que permite calcular el monto por penalización por superar el límite contratado (todos los métodos de todas las clases involucradas en conseguir esa funcionalidad), determine y enumere **qué métodos** testear, **indicando clases** y casos de prueba (teniendo en cuenta los conceptos de valores de borde y particiones equivalentes). Identifique claramente las particiones que encuentra y los valores de borde para cada caso.
- ii) Implemente utilizando JUnit los tests automatizados diseñados en el punto anterior.

4) **Instanciación**

- i) En un script de código o un test simple, escriba **el código Java** necesario para crear un cliente con un plan grupal con una cantidad máxima de 6 direcciones IP, que accedió a dos actividades: una reproducción de video, y una sesión de juegos, donde se utilizó un ítem de nombre “daño aumentado”, con 2 unidades y precio unitario de 1000. Luego el cliente cambia a un plan individual. Complete con los valores que necesite según corresponda.

Ejercicio 29 - Plataforma de Streaming

Instanciación

En un script de código o un test simple, escriba **el código Java** necesario para crear un cliente con un plan grupal con una cantidad máxima de 6 direcciones IP, que accedió a dos actividades: una reproducción de video, y una sesión de juegos, donde se utilizó un ítem de nombre “daño aumentado”, con 2 unidades y precio unitario de 1000. Luego el cliente cambia a un plan individual. **Complete con los valores que necesite según corresponda.**



FACULTAD DE INFORMATICA



UNIVERSIDAD
NACIONAL
DE LA PLATA

Ejercicio 29 - Plataforma de Streaming

- Lo importante es escribir **código Java** que cree los objetos que pide el enunciado, utilizando la solución que hayamos construido a lo largo del ejercicio
- Respetar y utilizar los constructores que hayamos definido en la solución. Recordar que todos los objetos deben proporcionar constructores para inicializarlos correctamente.
- Utilizar los mensajes que hayamos definido en nuestra solución para:
 - **Vincular los objetos**. Por ejemplo, si tenemos un mensaje “agregarActividad(param1, param2, paramN)”, debemos usarlo y **respetar los parámetros**, tal y cómo lo hayamos definido, con su orden y tipos.
 - Hacer trabajar a los objetos. Por ejemplo “*Luego el cliente cambia a un plan individual*”. Si nuestra solución define un mensaje para esto, deberíamos usar ese mensaje y no modificar una variable de instancia directamente



FACULTAD DE INFORMATICA



UNIVERSIDAD
NACIONAL
DE LA PLATA

Test visto en la explicación de práctica 5

```
1 public class PersonaTest {
2     private Persona james, guido;
3
4     @BeforeEach
5     void setUp() throws Exception {
6         james = new Persona();
7         guido = new Persona();
8         james.setApellido("Glosing");
9         james.setNombre("James");
10        guido.setApellido("van Rossum");
11        guido.setNombre("Guido");
12    }
13
14    @Test
15    public void testNombreCompleto() {
16        assertEquals("Glosing, James", james.getNombreCompleto());
17        assertEquals("van Rossum, Guido", guido.getNombreCompleto());
18    }
19 }
20 }
```

Creamos los objetos que vamos a probar

Verificamos los resultados esperados con "assertEquals"

¿Cómo sería un script de código?

```
1 Persona james = new Persona();  
2 Persona guido = new Persona();  
3 james.setApellido("Glosing");  
4 james.setNombre("James");  
5 guido.setApellido("van Rossum");  
6 guido.setNombre("Guido");
```

Con esto cumplimos la consigna. También es válido pero no necesario:

- Definir una clase con un método que contenga la instanciación de objetos
- Método main de Java que realice la instanciación

Parcial

- Sábado 8 de noviembre, 13:00 hs - Aulas 4, 5, 9, 10A y 11
 - Nos encontramos en el Aula 5 y luego nos vamos distribuyendo al resto
- Responder encuesta en Moodle, estará disponible desde el 3/11 00:00 hs hasta el 5/11 23:59hs



¿Asistiré al parcial del 8 de Noviembre?

Ocultado a los estudiantes

Debido a la situación presupuestaria que vivimos en las UUNN, queremos ser responsables al momento de generar las copias y no duplicar o triplicar la cantidad de enunciados necesarios.

Por esa razón, queremos saber si tenés pensado asistir al parcial del 8 de Noviembre para poder estimar la cantidad de enunciados a imprimir. Te pedimos que completes esta consulta con total honestidad, es anónima y si cambiás de opinión y te presentás o no te presentás el sábado, no pasa nada.

Parcial

- El examen es en papel
 - Evalúa todos los contenidos de la materia que vimos en la práctica, en el cuadernillo de actividades que trabajamos a lo largo de la cursada
 - El **único** apunte que se puede utilizar en el parcial es el documento “Cheatsheet de Streams”
- **Ningún** otro apunte está permitido en el parcial
 - No es necesario aprenderse los imports de Java de memoria
 - No es necesario escribir todos los getter y setter
 - Sólo los que nuestra solución requiera
 - Recordar encapsulamiento y delegación. ¿Es necesario ese getter/setter que quiero definir o solamente lo hago por mecánica?

API STREAM	
FILTER Filtra los elementos de un stream según el predicado que recibe como parámetro. Ej: obtener los alumnos que ingresaron en un año dado	ese caso retorna Ej: consultar si al o no. public boolean e return alumnos .anyMatch(alum
<pre>public List<Alumno> ingresantesEnAño(int año){ return alumnos.stream() .filter(alumno->alumno.getAñoIngreso() == año) .collect(Collectors.toList());</pre>	Op. relacionadas: Evalúa si todos lo del stream satis parámetro. y e

Foros de consulta

Cómo preguntar en el foro

Breve guía para poder sacar el mejor provecho al foro y a la convivencia a través de las preguntas y respuestas.

[Cómo preguntar en el foro](#)

[Antes de Preguntar: Busca una respuesta por tus propios medios](#)

[Elegí el foro específico](#)

[Elegí un título apropiado para la pregunta](#)

[No envíes una solución para que la corrijan](#)

[Describir qué estás intentando hacer](#)

[Describir el problema y lo que has intentado para resolverlo](#)

[Escribir claro](#)

[No solicites respuestas a tu correo](#)

[Si no entendés la respuesta](#)

[Terminá con una breve nota de conclusión.](#)

[Evitá el "Me sumo al pedido"](#)



Foro: Ejercicio 12 - Job Scheduler



Foro: Ejercicio 13 - ¡A implementar Inversores!



Foro: Ejercicio 14 - Volumen y superficie de sólidos



FACULTAD DE INFORMATICA



UNIVERSIDAD
NACIONAL
DE LA PLATA