

**Universidad Tecnológica Nacional  
Facultad Regional Avellaneda**



Técnico Superior en Programación - Técnico Superior en Sistemas Informáticos										
Materia: Laboratorio de Programación II										
Apellido:					Fecha:	13/12/2018				
Nombre:					Docente <sup>(2)</sup> :					
División:					Nota <sup>(2)</sup> :					
Legajo:					Firma <sup>(2)</sup> :					
Instancia <sup>(1)</sup> :	<b>PP</b>	<input checked="" type="checkbox"/>	<b>RPP</b>	<input type="checkbox"/>	<b>SP</b>	<input type="checkbox"/>	<b>RSP</b>	<input type="checkbox"/>	<b>FIN</b>	<input type="checkbox"/>


(1) Las instancias validas son: 1<sup>er</sup> Parcial (**PP**), Recuperatorio 1<sup>er</sup> Parcial (**RPP**), 2<sup>do</sup> Parcial (**SP**), Recuperatorio 2<sup>do</sup> Parcial (**RSP**), Final (**FIN**). Marque con una cruz.

(2) Campos a ser completados por el docente.

**IMPORTANTE:**

- Guardar el proyecto en el **disco D:**. Ante un corte de energía o problema con el archivo de corrección, el alumno será responsable de que el proyecto sea recuperable.
- **2 (dos) errores en el mismo tema anulan su puntaje.**
- **Errores de conceptos de POO anulan el punto.**
- **Cada tema vale 1 (un) punto (Herencia, Generics, Test Unitarios, etc.). La correcta documentación también será evaluada.**
- **Se deberán tener al menos el 60% bien de los temas a evaluar según la instancia para lograr la aprobación.**
- Colocar sus datos personales en el nombre del proyecto principal, colocando: Apellido.Nombre.AñoCursada. Ej: Pérez.Juan.2019. No se corregirán proyectos que no sea identificable su autor.
- **Salvo que se indique lo contrario, TODAS** las clases deberán ir en una Biblioteca de Clases llamada Entidades.
- No se corregirán exámenes que no compilen.
- **Reutilizar** tanto código como crean necesario.

Al finalizar, colocar la carpeta de la Solución completa en un archivo ZIP que deberá tener como nombre

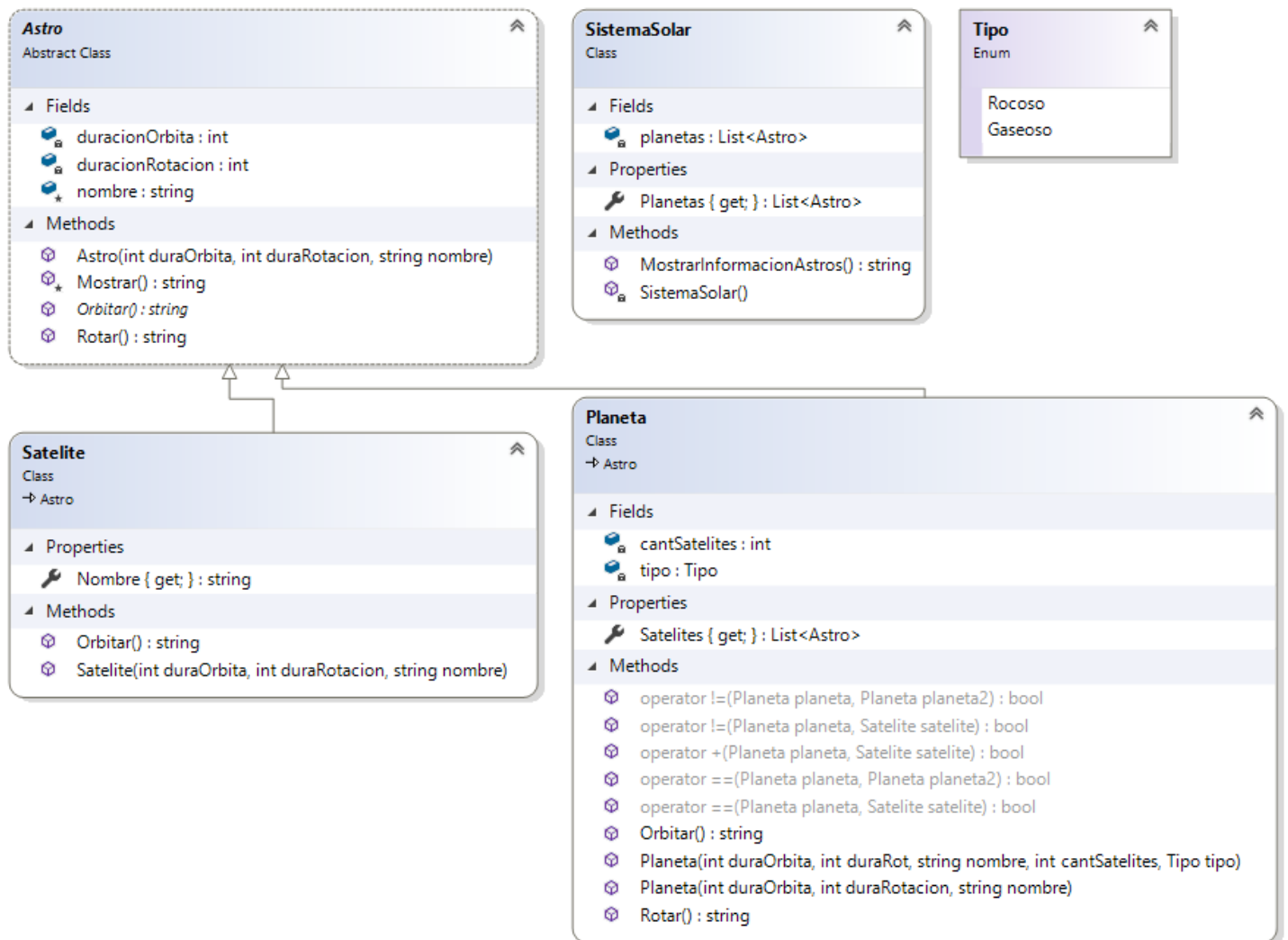
Apellido.Nombre.AñoCursada.zip y dejar este último en el Escritorio de la máquina. Luego presionar el botón  de la barra superior, **colocar un mensaje** y presionar *Aceptar*. **Aguardar a que el profesor indique que el examen fue copiado de forma correcta.** Luego retirarse del aula.

---

*TIEMPO MÁXIMO PARA RESOLVER EL EXAMEN 110 MINUTOS.*

---

1. Generar un proyecto llamado Entidades y colocar dentro el siguiente esquema de clases:



#### Astro:

- Clase abstracta con un constructor que recibe su nombre y la duración de órbita y la de rotación.
- Mostrar() es un método protegido que devuelve toda la información del astro. (Ej: Nombre: {nombre} - Órbita: {órbita} -Rotación: {rotación})
- Orbitar() método abstracto.
- Rotar() método virtual que retorna el mensaje "Rotando. Tiempo estimado: {tiempo de rotación}."
- Agregar un conversor explícito de Astro a String que retorne sólo el nombre del astro.

#### Tipo:

- Enum con dos valores: Rocoso y Gaseoso.

#### Planeta:

- Hereda de Astro y agrega 3 atributos privados: cantidaSatelites (int), tipo (Tipo), y satélites (List<Astro>).
- Tiene una propiedad que retorna la lista de satélites.
- Tiene un constructor que recibe los mismos parámetros que el constructor de Astro y otro que agrega cantSatelites y el tipo.
- Sobrecarga el operador + para agregar un satélite a la lista, como recibe un objeto del tipo Astro debe validar que sea satélite y no planeta.
- Sobre carga del == (Planeta, Satélite) que chequea si el satélite se encuentra en la lista (comparando el nombre).
- Sobre carga del == (Planeta, Planeta) compara dos planetas por el nombre.
- Orbitar() retorna el siguiente mensaje "Orbita el planeta: {nombre}".
- Rotar() método que no sobrescribe el base. Retorna: "Orbita el planeta {nombre}"
- Sobrecarga del ToString() que retorna la información del objeto.

### Satélite:

17. Hereda de Astro y agrega la propiedad que retorna el nombre.
18. Orbitar() retorna "Orbitar el satélite: {nombre}".
19. Sobrecarga del ToString() que retorna la información del objeto.

### SistemaSolar:

20. Planetas: atributo privado del tipo List<Astros>.
21. Posee un solo constructor sin parámetros.
22. MostrarInformacionAstros(): método que retorna toda la información de los planetas, y sus satélites.

### FormParcial:

23. Generar el siguiente formulario:

The image displays a Windows Forms application. On the left, the 'FormSistemaSolar' class is shown in the Solution Explorer, listing its fields and methods. The fields include buttons for adding planets and satellites, a list box for planets, a dropdown for planet type, and various numeric variables for orbital and rotational data. The methods include click events for the buttons and a load event for the form. On the right, the 'Nombre Alumno' form is shown, which contains a grid of input fields for planet and satellite information, including names, orbital completion times, rotational completion times, and the number of moons. The form also features buttons for 'Agregar Planeta', 'Agregar Satelite', 'Mostrar Informacion', and 'Mover Astros'.

24. Posee una atributo estático, planetas, el cual es del tipo List<Astro>
25. Posee un constructor de instancia.
26. Se debe validar que no se ingresen campos vacíos al agregar un planeta.
27. Se debe validar que en el tiempo en completar órbita se introduzca un número positivo.
28. La lista de Tipo de Planeta debe tener los dos tipos de planetas.
29. Se debe validar que no se ingresen campos vacíos al agregar un satélite.
30. La lista de planetas debe actualizarse cuando se agrega un planeta (con los nombres de los mismos).
31. Se debe validar que haya planetas y se haya elegido uno al momento de agregar un satélite.
32. Mostrar información muestra la información de todos los planetas y satélites en el rich text box.
33. Mover Astros ejecuta el método Orbitar y Rotar de todos los astros. En caso de los Planetas se debe ejecutar el método Rotar de Planeta (hijo) y no de Astro (padre).