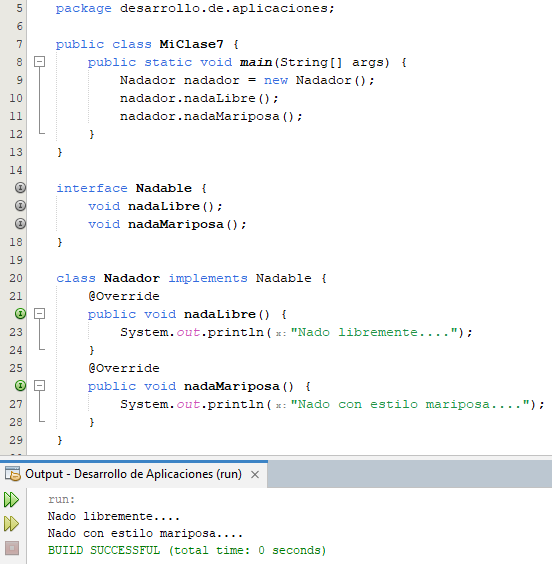
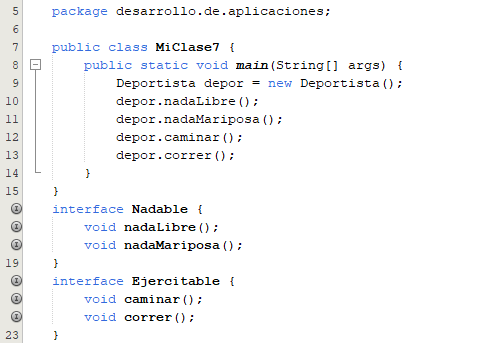
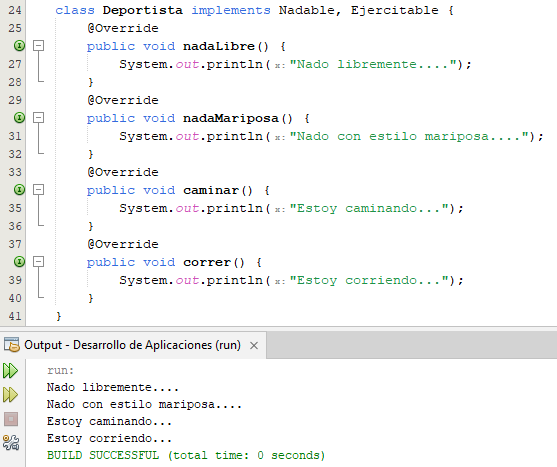
**Desarrollo Avanzado de Aplicaciones I**

**Ejercicio 1**. Desarrollar una aplicación que defina una interface “Nadable” que tenga dos métodos abstractos: “nadaLibre” y “nadaMariposa”. Debe definir también una clase “Nadador” que implemente la interface “Nadable” y que sobrescriba sus dos métodos. Definir una clase con método main que use la clase “Nadador” y que invoque a sus dos métodos.

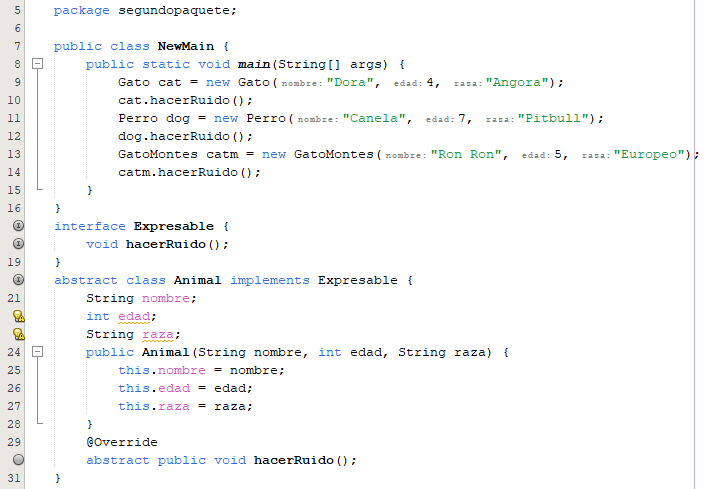


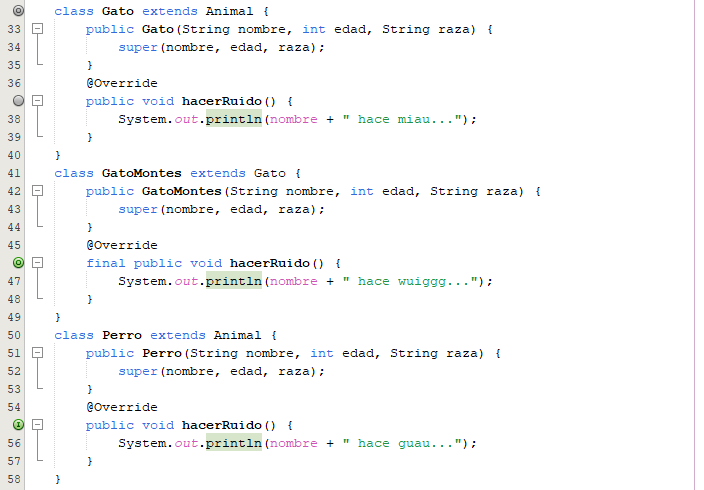
**Ejercicio 2**. Desarrollar una aplicación que defina dos interfaces: “Nadable” y “Ejercitable”. La interface “Nadable” debe tener dos métodos abstractos: “nadaLibre” y “nadaMariposa” y la interface “Ejercitable” debe tener los métodos abstractos “caminar” y “correr”. Debe definir también una clase “Deportista” que implemente las interfaces “Nadable” y “Ejercitable” y que sobrescriba los cuatro métodos. Definir una clase con método main que use la clase “Deportista” y que invoque a sus métodos.

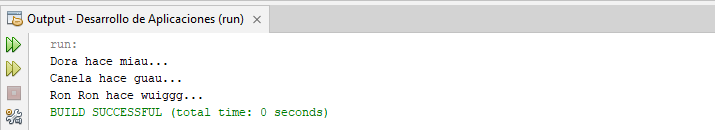




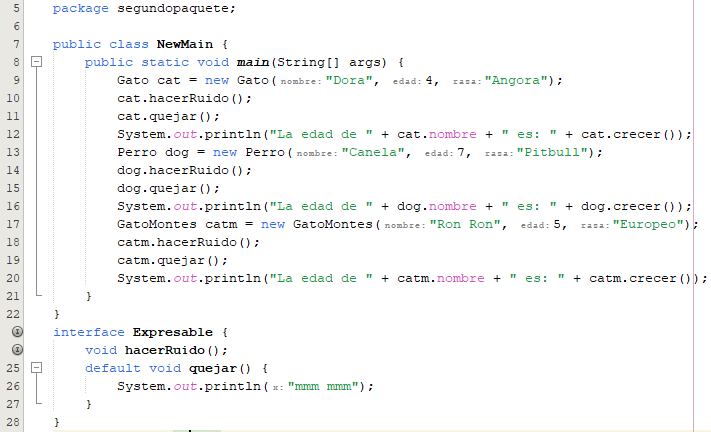
**Ejercicio 3**. Desarrollar una aplicación que defina una interface “Expresable” que tenga un método “hacerRuido”. Definir una clase abstracta “Animal” que implemente la interface “Expresable” y que tenga los atributos “nombre”, “edad”, “raza” y “hacerRuido”. El método “hacerRuido” debe ser abstracto. Definir las clases “Gato” y “Perro” que sean derivadas de “Animal” y que sobrescriban el método “hacerRuido” con sus propios sonidos. Definir una clase “GatoMontes” que sea derivada de la clase “Gato” y que también sobrescriba el método “hacerRuido” con su propio sonido.

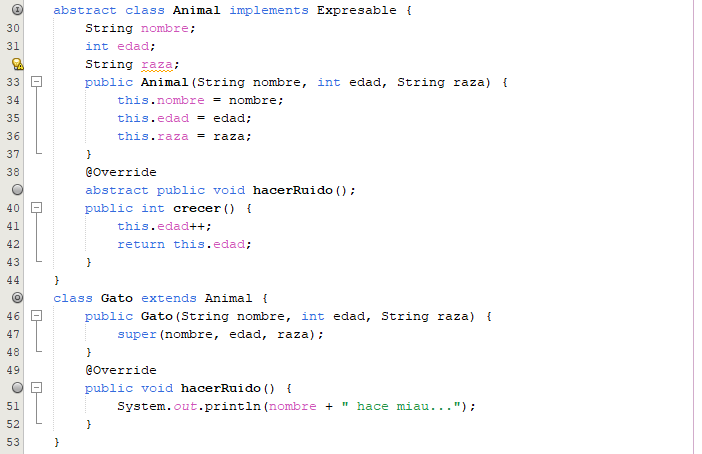


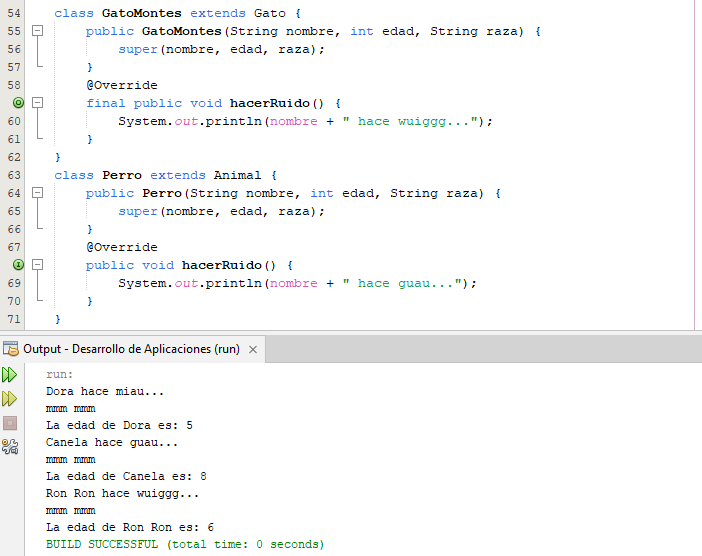




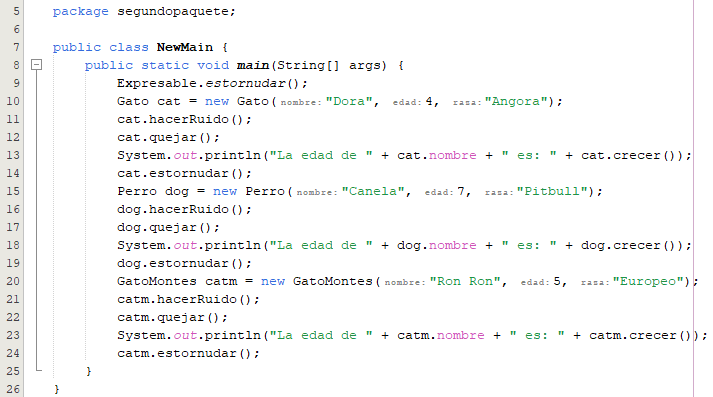
**Ejercicio 4**. Modificar el ejercicio anterior para que la interface “Expresable” tenga un método por defecto (default) llamado “quejar” que muestre un mensaje de queja (“mmm mmm”). A la clase “Animal” agregar un método “crecer” que incremente en 1 la edad del animal y que devuelva la edad del animal. Agregar al método main la invocación a los métodos nuevos. Discutir la diferencia de utilizar los tipos de métodos agregados.

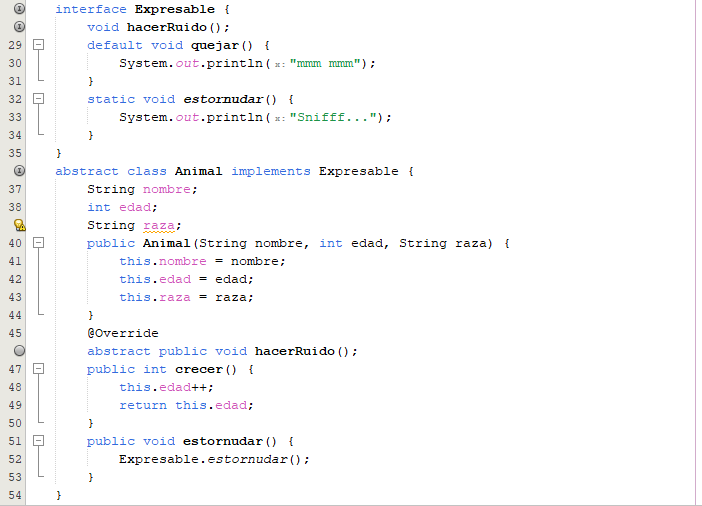




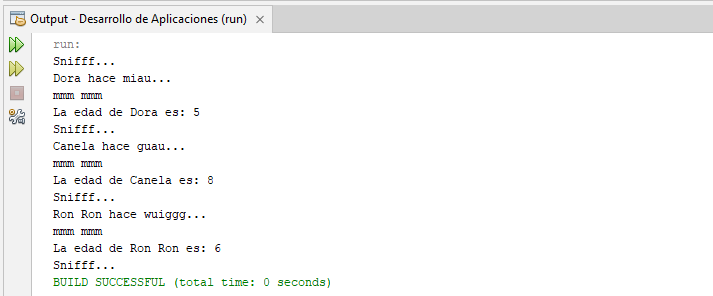


**Ejercicio 5**. Modificar el ejercicio anterior agregando un método estático a la interface “Expresable” que se llame “estornudar” y que muestre el mensaje “Snifff…”. Agregar un método “estornudar” a la clase “Animal” y que invoque al método estático de la interface. Agregar al método main una invocación directa al método de la interface y llamadas al método estornudar de cada clase derivada.

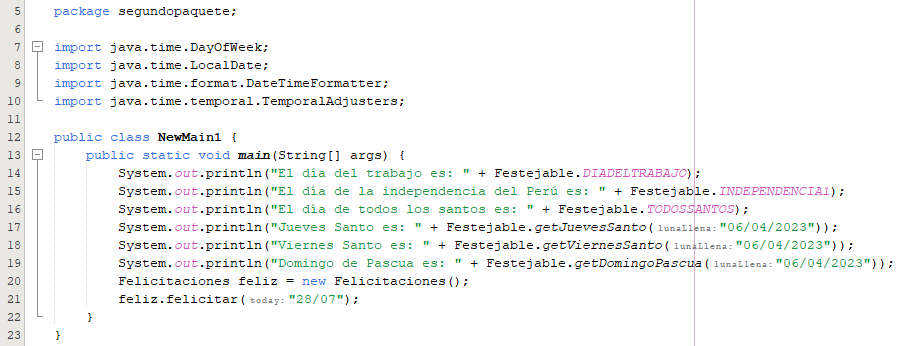


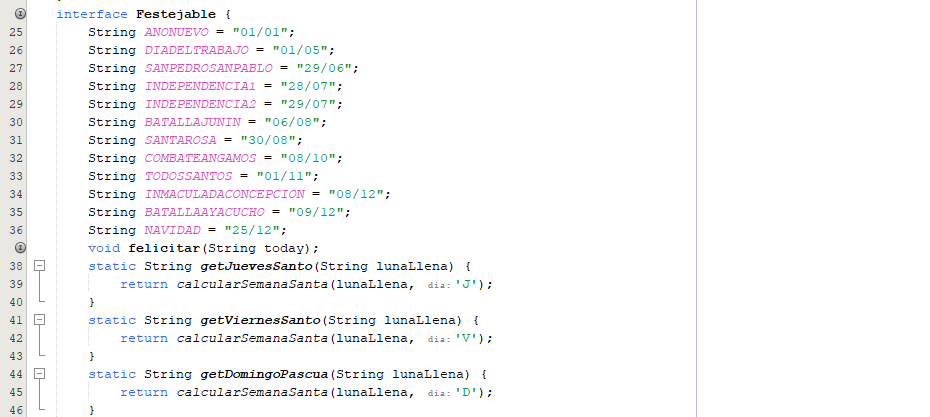


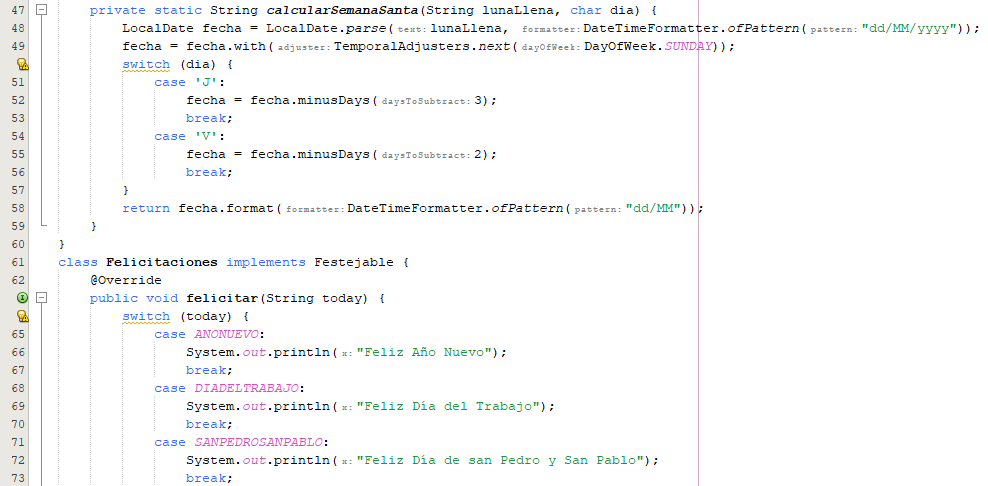
Las clases “Gato”, “Perro” y “GatoMontes” no se modifican.

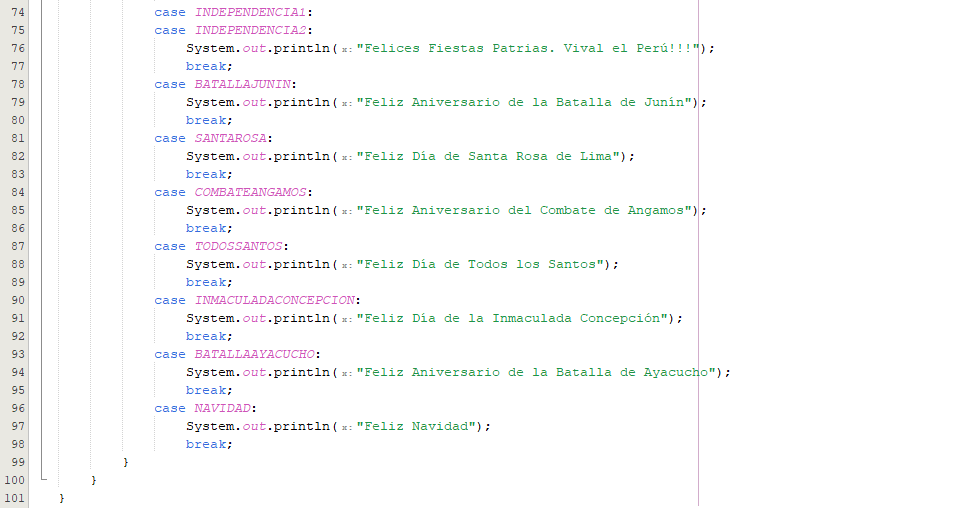


**Ejercicio 6**. Desarrollar una aplicación que defina una interface “Festejable” que tenga como atributos constantes los feriados fijos del año. La interface debe tener un método abstracto “felicitar” que reciba una fecha en formato “dd/MM”. Adicionalmente debe tener 3 métodos estáticos que te devuelvan los feriados de semana santa. Estos métodos deben recibir como parámetro la fecha de la siguiente luna llena después del 21 de marzo y llamar a un método privado que calcule y devuelva la fecha del día de semana santa, recibiendo como parámetros la misma fecha de la luna llena y la inicial del día de semana santa a calcular. Luego definir una clase “Felicitaciones” que implemente la interface “Festejable” y que sobrescriba el método “felicitar” comparando la fecha que recibe como parámetro con los feriados que proporciona la interface, mostrando un mensaje en pantalla de felicitaciones de acuerdo a cada día. Definir un método main que use los atributos constantes de la interface y sus métodos estáticos. También debe instanciar la clase e invocar al método “felicitar”.

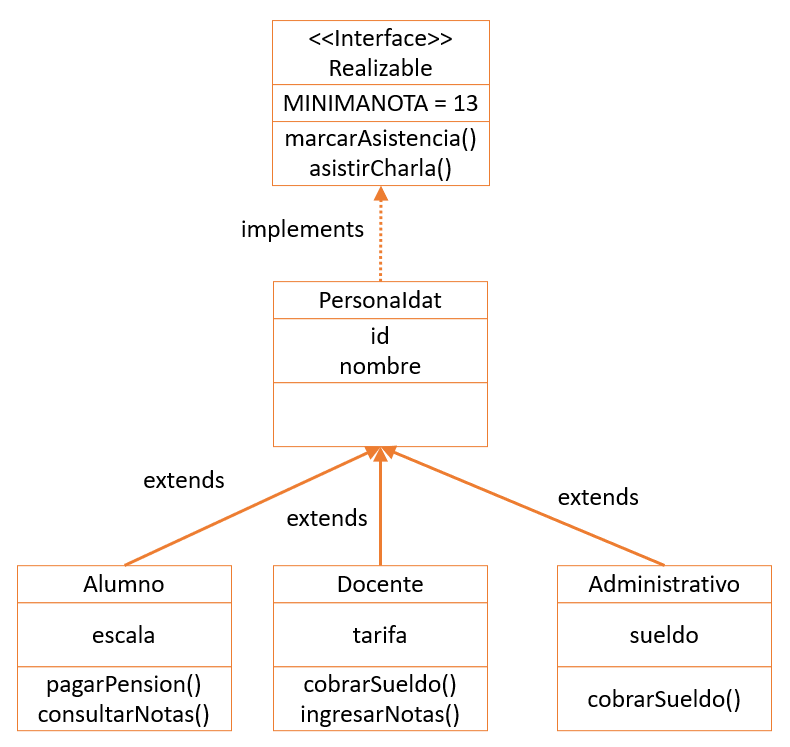






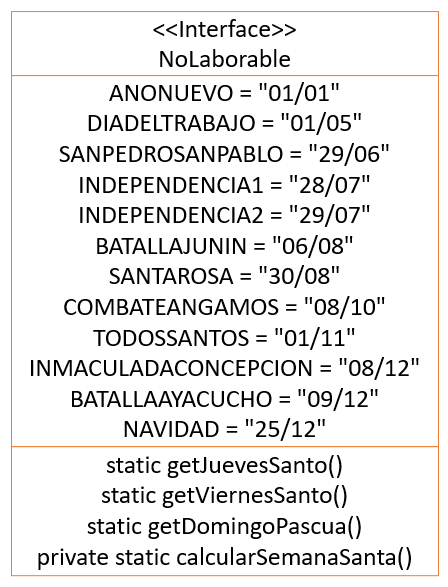


**Ejercicio 7**. Desarrollar una aplicación que implemente la representación de la siguiente imagen:



Los métodos de las clases solo implementan el mostrar un mensaje de acuerdo al nombre del método.

**Ejercicio 8**. Modificar el ejercicio anterior y agregarle la interface NoLaborable para que se puedan hacer los cálculos de pagos y se controlen las asistencias, como se muestra en la siguiente imagen:



Se debe probar el agregar la interface de tres maneras diferentes:

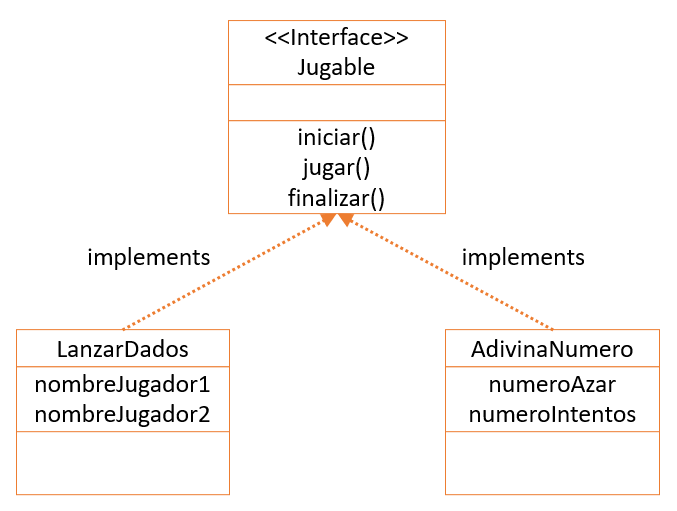
1. La clase “PersonaIdat” implementa las dos interfaces “Realizable” y “NoLaborable”.
2. La interface “Realizable” hereda de la interface “NoLaborable”.
3. Las clases “Alumno”, “Docente” y “Administrativo” implementan la interface “NoLaborable”.

**Ejercicio 9**. Desarrollar una aplicación que permita jugar dos juegos distintos:

El primer juego se llama “LanzarDados” y consiste en que dos jugadores tiren un dado y el número mayor gana. El juego se inicia ingresando el nombre de los jugadores. Este juego se realiza tirando el dado por cada jugador y mostrando en pantalla que número sacó. El juego finaliza mostrando cual de los jugadores ganó.

El segundo juego se llama “AdivinaNumero” y consiste en que un jugador debe adivinar un número que elige la computadora al azar en el rango del 1 al 100. El juego se inicia generando el número al azar. Este juego se realiza solicitando que el jugador adivine (ingrese por teclado) el número al azar que eligió el computador, se debe contabilizar el número de intentos que ha realizado el jugador hasta adivinar el número. Cada vez que el jugador ingrese un número se le debe indicar si el número a adivinar es menor o mayor al número que ingresó. El juego finaliza cuando el jugador adivinó el número y se debe mostrar el número de intentos que realizó.

Podemos decir que todo juego tiene un inicio, el juego propiamente dicho y un fin. Debemos estandarizar estos dos juegos declarando una interface con nombre “Jugable” que declare los métodos: “iniciar”, “jugar” y “finalizar” y las clases (los juegos) deben implementarla. Se tiene como referencia la siguiente imagen:



Crear un método main donde se pueda jugar los juegos.