

```
class Vehiculo:
```

```
|   Vehiculo   | <--  
+-----+  
| + moverse() |  
| + detenerse() |  
+-----+
```

```
def moverse(self):
```

```
|   Coche    | |Bicicleta|  
+-----+ +-----+  
| + moverse() | + moverse()  
| + detenerse() | + detenerse()  
+-----+ +-----+
```

Proyecto: El juego de blackjack



Proyecto: El juego de blackjack

El siguiente documento ha sido elaborado tomando fragmentos de documentos previamente publicados, así como imágenes y ejercicios, sobre los cuales se han realizado traducciones libres y adaptaciones con el propósito de desarrollar el material para el curso de Fundamentos de Programación Orientada a Objetos de la Universidad del Valle. Se ha respetado la autoría original de los materiales, y se han proporcionado las referencias correspondientes dentro del documento. El uso de este material se restringe estrictamente al marco académico y formativo de los estudiantes que cursen la asignatura, y no tiene fines comerciales. Queda prohibida su copia o distribución fuera del contexto de dicho curso. El proyecto fue tomado de [1] se hizo una traducción libre y se adaptó al curso.

Entrega Final: Blackjack



Llegó el momento de la entrega final de nuestro proyecto "Blackjack" donde trabajaremos en afinar el diseño del sistema, a través del uso de una clase controladora o un patrón de diseño adecuado. Además, de implementar pruebas unitarias para garantizar la calidad del código y actualizar la documentación de las clases. Veamos los pasos a seguir:

2 Fundamentos de programación orientada a objetos

Proyecto: El juego de blackjack

Entrega final del Proyecto

1. Analizar el diagrama de clases del proyecto ver 2.0 y su implementación, verificar que se aplica una clase controladora o un patrón de diseño. Si no encuentra la aplicación del concepto en el diagrama de clases del proyecto y su implementación, corregir.
2. Implementación de pruebas unitarias: utilice *assert*, *try* o *google test* para cada uno de los métodos del proyecto *Blackjack*.
3. De acuerdo a su diagrama de clases del proyecto ver 3.0 entregar la implementación de las definiciones de las clases con su respectiva documentación (archivos.h).
4. De acuerdo a su diagrama de clases del proyecto ver 3.0 entregar la implementación de las clases con su respectiva documentación (archivos.cpp).
5. Entregar la **versión final** del programa. que demuestre que la implementación del diagrama de clases actualizado funciona correctamente y cumple con los requisitos del proyecto.

Referencias

- [1] M. A. Weisfeld, *The object-oriented thought process*, 3rd ed. en Developer's library. Upper Saddle River, NJ: Addison-Wesley, 2009.
- [2] J. A. Villalobos S., *Fundamentos de programación: aprendizaje activo basado en casos : un enfoque moderno usando Java, UML, objetos y eclipse*. Colombia: Pearson, 2006.