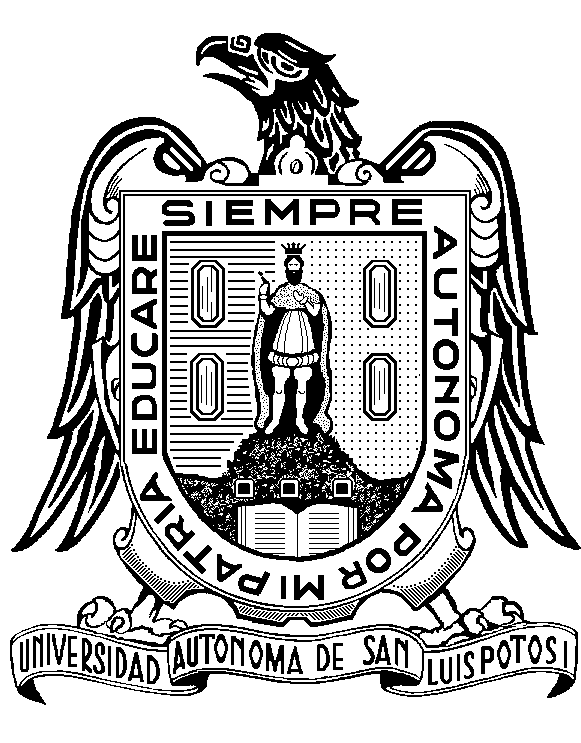
Universidad Autónoma de San Luis Potosí

Facultad de Ingeniería



[Área de Computación e Informática](http://infocomp.ingenieria.uaslp.mx/sistemaS2.1/)

Manual del programador

No. de proyecto: 179

Proyecto: MEGAMAN

Alumno: Ángel Modesto Hernández Méndez

Objetivo:

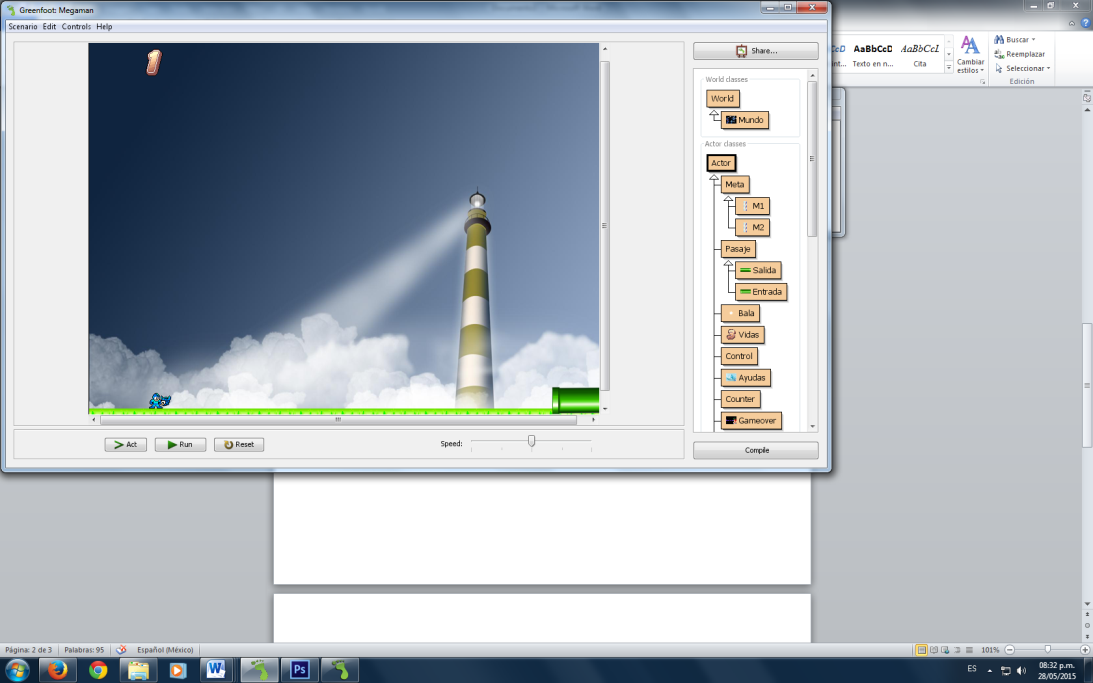
Implementar y aplicar los conocimientos aprendidos durante el curso de programación orientada a objetos.

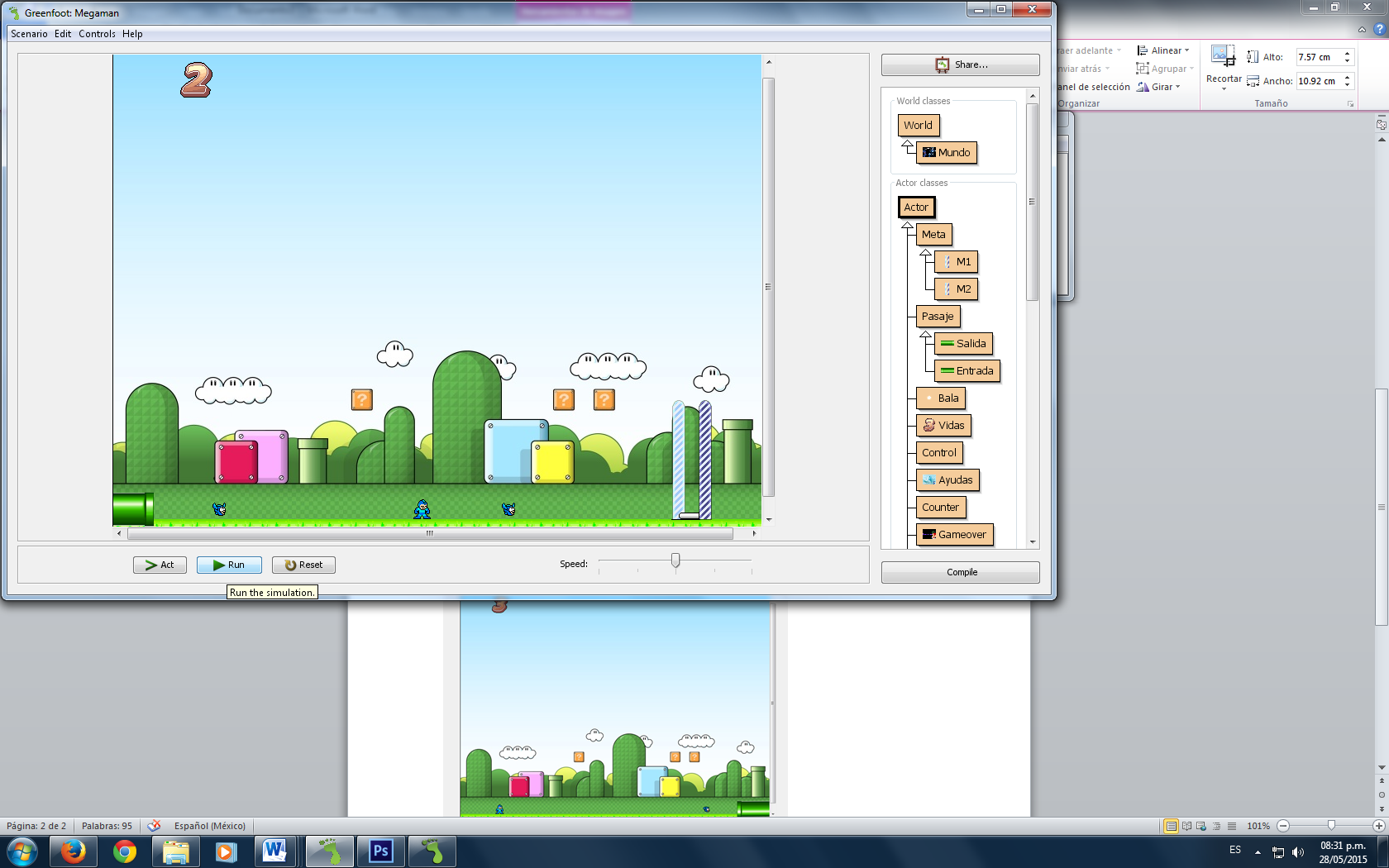
Descripción:

El juego consiste en que el personaje principal de éste, llamado Megaman; se encuentra atrapado en el mundo de Mario Bros, pasando por obstáculos típicos del mundo de Mario Bros y enemigos de Megaman.

En el juego se tendrán tres vidas por nivel.

Megaman tendrá enemigos o adversarios que intentaran impedir que logre su misión.

1er Nivel 2do Nivel



3er Nivel

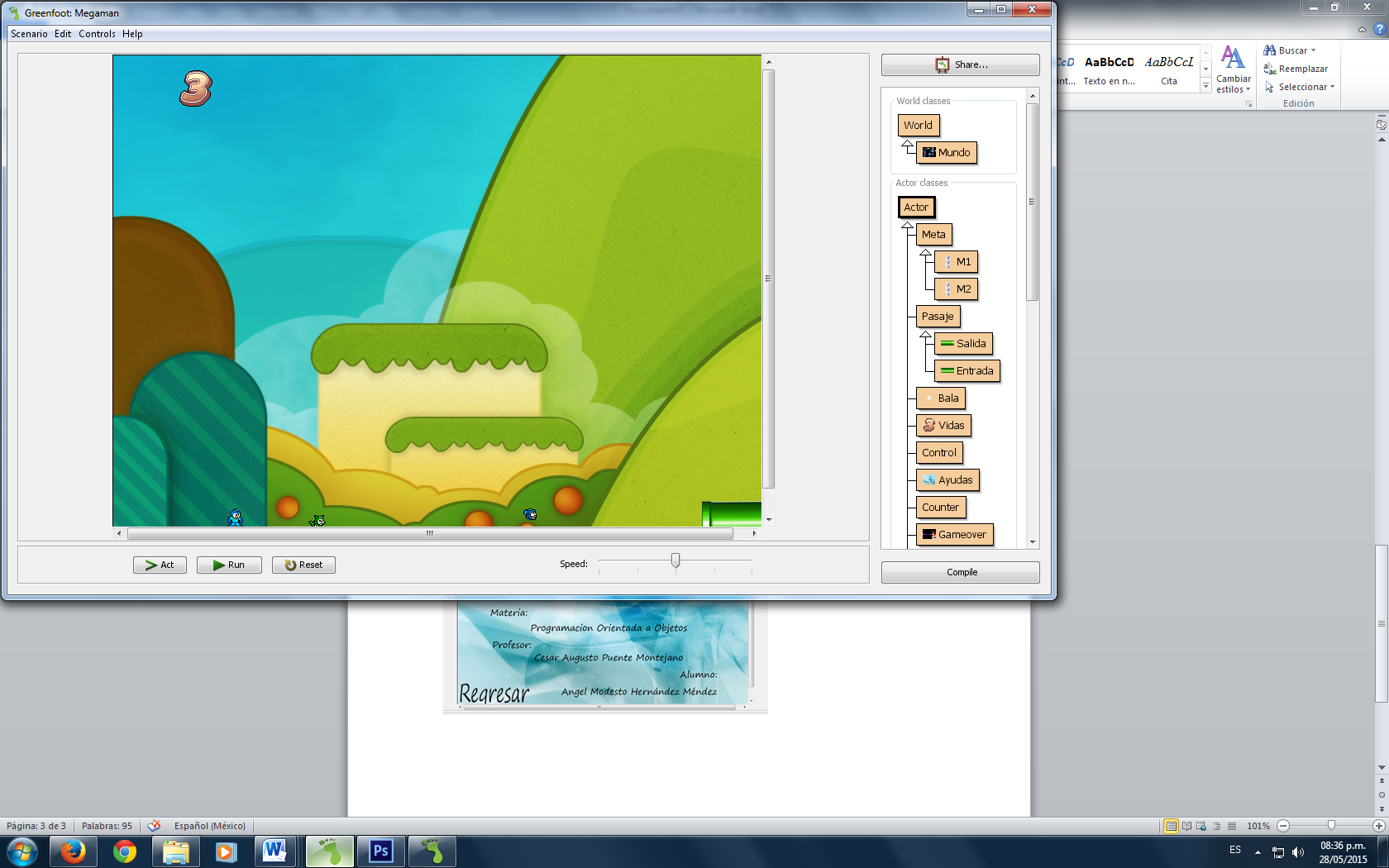
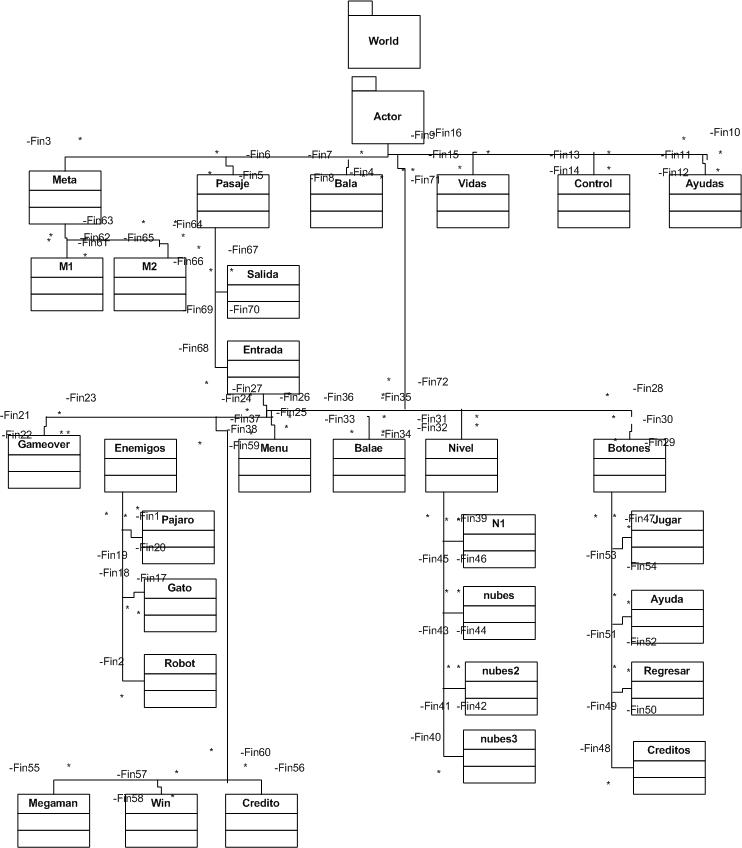


Diagrama de clases



|  |
| --- |
| \*public class **Meta**extends greenfoot.Actor |
| La clase Meta es la clase que tendra como hijos las dos imagenes que se necesitaran para que megaman pueda pasar de nivel |

|  |
| --- |
| \*public class **M1**extends Meta |
| }La clase M1 nos muestra el objeto que se vera de la meta antes de que el jugador llegue a esta |

|  |
| --- |
| \*public class **M2**extends Meta |
| La clase M2 sirve para mostrar la meta despues de que el jugador pasa por el objeto de la clase M1 |

|  |
| --- |
| \*public class **Pasaje**extends greenfoot.Actor |
| Metodo que trasnporta de una parte del escenario a otra |

|  |
| --- |
| \*public class **Salida**extends Pasaje |
| La clase Salida nos mostrara el tubo donde megaman saldra despues de pasar de nivel |

|  |
| --- |
| \*public class **Entrada**extends Pasaje |
| La clase Entrada nos mostrara el tubo que megaman usara para pasar escenario |

|  |
| --- |
| \*public class **Bala**extends greenfoot.Actor |

|  |
| --- |
| \*public class **Vidas**extends greenfoot.Actor |
| Clase donde se modificara y mostrara el contador de las vidas del jugador |

|  |
| --- |
| \*public class **Control**extends greenfoot.Actor |
| Clase donde se lleva el control de todo lo que sucede en el escenario, niveles, puntos, etc. |
| **cambiaN1** |
| public void **cambiaN1**() |
| En este metodo se quitan todos los objetos que existen en el primer nivel para poder cargar la segunda parte |
|  |
| **cambiaN2\_p2** |
| public void **cambiaN2\_p2**() |
| Quita los objetos que se encontraban en la primera parte del segundo nivel |
|  |
| **cambiaN2** |
| public void **cambiaN2**() |
| Metodo que quita los objetos para pasar al nivel 2 |
|  |
| **cambiaN3\_p2** |
| public void **cambiaN3\_p2**() |
| Retira los obejtos de la primra parte del nivel 3 |
|  |
| **cambiaN3** |
| public void **cambiaN3**() |
| Quita los objetos del nivel 3 |
|  |
| **CargaN1\_p1** |
| public void **CargaN1\_p1**() |
| Aqui se carga el nivel 1 |
|  |
| **CargaN1\_p2** |
| public void **CargaN1\_p2**() |
| Aqui se carga la segunda parte del nivel 1 |
|  |
| **CargaN2\_p1** |
| public void **CargaN2\_p1**() |
| Metodo que inicializa todos los objetos para el segundo nivel |
|  |
| **CargaN2\_p2** |
| public void **CargaN2\_p2**() |
| Coloca los obejtos de la segunda parte del nivel 2 |
|  |
| **CargaN3\_p1** |
| public void **CargaN3\_p1**() |
| Carga los objetos de la primra parte del nivel 3 |
|  |
| **CargaN3\_p2** |
| public void **CargaN3\_p2**() |
| Coloca los objetos de la segunda parte del nivel 3 |
|  |
| **ContadorBandera** |
| public void **ContadorBandera**() |
| Este metodo sirve para cambiar el escenario |
|  |
| **gameOver** |
| public void **gameOver**() |
| Metodo que manda a llamar a la clase Gameover que es donde se encuentra la imagen que nos cambiara al aperder todas las vidas |

|  |
| --- |
| public class **Ayudas**extends greenfoot.Actor |
| Clase donde se contiene la imagen de ayuda asi como se coloca el boton para regresar al menu |

|  |
| --- |
| public class **Gameover**extends greenfoot.Actor |
| Clase donde se contiene la imagen de GameOver |

|  |
| --- |
| public class **Enemigos**extends greenfoot.Actor |
| Clase donde se encontraran todos los enemigos del juego |

|  |
| --- |
| public class **Pajaro**extends Enemigos |
| Clase pajaro en la cual se crearan todos los metodos que hacen que se mueva |
| **compruebaObjetivo** |
| public void **compruebaObjetivo**() |
| En este metodo se comprueva que durante el movimiento realizado por el enemigo no tenga contacto con el jugador megaman y si es asi se disminuyen vidas |
|  |
| **Move** |
| public void **Move**() |
| Metodo que deacuerdo a la direccion que recibe le indica al enemigo hacia que lado debera de moverse |
|  |
| **Walk** |
| public void **Walk**(int dir) |
| Metodo que hace que el enemigo pajaro tenga una secuencia de movimiento |
|  |

|  |
| --- |
| public class **Gato**extends Enemigos |
| Clase gato en la cual se crearan todos los metodos que hacen que se mueva |
| **compruebaObjetivo** |
| public void **compruebaObjetivo**() |
| En este metodo se comprueva que durante el movimiento realizado por el enemigo no tenga contacto con el jugador megaman y si es asi se disminuyen vidas |
|  |
| **Move** |
| public void **Move**() |
| Metodo que deacuerdo a la direccion que recibe le indica al enemigo hacia que lado debera de moverse |
|  |
| **Walk** |
| public void **Walk**(int dir) |
| Metodo que hace que el enemigo pajaro tenga una secuencia de movimiento |

|  |
| --- |
| public class **Robot**extends Enemigos |
| Clase robot en la cual se crearan todos los metodos que hacen que se mueva |
| **compruebaObjetivo** |
| public void **compruebaObjetivo**() |
| En este metodo se comprueva que durante el movimiento realizado por el enemigo no tenga contacto con el jugador megaman y si es asi se disminuyen vidas |
|  |
| **dispara** |
| public void **dispara**(int x, |
| int y, |
| int dir) |
| Metodo que agrega un objeto de tipo bala que sera usada contra el jugador |
|  |
| **Move** |
| public void **Move**() |
| Metodo que deacuerdo a la direccion que recibe le indica al enemigo hacia que lado debera de moverse |
|  |
| **Walk** |
| public void **Walk**(int dir) |
| Metodo que hace que el enemigo pajaro tenga una secuencia de movimiento |

|  |
| --- |
| public class **Megaman**extends greenfoot.Actor |
| En esta clase se contienen todos los movimientos y acciones relacionadas con el jugad |
| **aumentaDano** |
| public void **aumentaDano**() |
| Metodo que aumenta el daño y asi reduce las vidas del jugador |
|  |
| **dispara** |
| public void **dispara**(int x, |
| int y, |
| int dir) |
| Metodo que hace que se proyecte una bala que pertenece al jugador |
|  |
| **isLive** |
| public boolean **isLive**(int hpoints) |
| Metodo que realiza un chequeo de las vidas que tiene y el daño que a recivido para determinar si sigue con vida o el juego termino |
|  |
| **Move** |
| public void **Move**() |
| En este metodo se realizan las lecturas de teclas de direcciones para determinar el sentido de las imagenes y asi mover el jugador en la direccion correcta |
|  |
| **MoveAux** |
| public void **MoveAux**(int x, |
| int y, |
| int dir) |
| Metodo que realiza la secuencia de imagenes para que se vea la animacion correcta en la direccion recivida |
|  |
| **salta** |
| public void **salta**() |
| Metodo que hace saltar al jugador y regresarlo a la plataforma donde lo reealizo |
|  |

1. Herencia y polimorfismo:

En el proyecto la herencia fue utilizada para poder agrupar a todos los enemigos siendo todos de este tipo, pero diferentes enemigos con acciones particulares, también para poner los botones, ya que actuaran dirigiendo hacia un lugar en el juego pero con diferentes direcciones y nombres.

El polimorfismo se usa al momento de enviar desde la clase enemigo a la clase megaman que ha sufrido daño porque este le ha tocado o en su caso disparado, también para el jugador al momento de presionar la tecla espaciadora poder disparar dándole la orden al objeto bala para que a su vez pueda checar si a impactado algo en el campo de juego.

Etapas de la Elaboración del Proyecto:

>Una de las partes iniciales del proyecto es comenzar con la redacción de la idea de nuestro proyecto, en este caso la concepción de la idea del juego.

>La idea es aprobada y revisada por el profesor y una vez que es aprobada se comienza la implementación de las clases del programa.

>A continuación se empieza a trabajar en la elaboración de la Portada, los menús de Ayuda y la Introducción a la Clase Juego.

>Se buscan y ajustan las imágenes de acuerdo a la temática de nuestro juego: personajes, entorno y obstáculos.

>Posteriormente se acomodan dichas imágenes; se realizan recortes y se comienza con la implementación del movimiento del jugador.

(Dentro de este punto se presentó dificultad en cuanto a la implementación del salto del personaje principal para esquivar a sus enemigos, lo cual engloba de forma general un problema de manejo de la gravedad dentro del campo de juego)

>Continuando con el proceso de elaboración del juego se reescribe el código de movimiento del jugador y se copea a las Clases Enemigos.

>Después se comienzan a colocar los obstáculos dentro del campo de juego, se acomodan los enemigos en diversos sitios estratégicos para que aparezcan y ataquen al jugador.

>Se implementan objetos de tipo bala tanto para el enemigo como para el jugador.

>Se crean los Niveles que el jugador debe de superar y se ajustar el contador de vidas.

>Se crearon los textos de Ayuda y Créditos que se agregaron en el menú.

>Se checa y prueba el producto terminado.

Link Greenfoot

<http://www.greenfoot.org/scenarios/14049>