CSE1042 Programming Project
(Project Niyazi)
(Alpha Version)

11/04/2016
Marmara University
Faculty of Engineering
Computer Engineering

Supervisor:

Associate Professor Ali Fuat ALKAYA
Senior Lecturer
Marmara University Faculty of Engineering Computer Engineering Department
E-mail: falkaya@gmail.com - falkaya@gmail.com

Supervisee:

Bilgehan NAL
Student of Computer Engineering Department at Marmara University
Student Id: 150114038
E-Mail: bilgehanl.03@gmail.com

Abstract

In this project I had tried to create a kind of library which is getting easier the creating a 2D platform game for Java. After import the Project Niyazi to a Java Project, user can create a game with less information and less calculation. Also User who has information of some frameworks or another technologies can create different projects combining the other technologies and Project Niyazi.
In addition user can shape the Project Niyazi by himself/herself downloading the source code of the project.

This version (Alpha Version) of the project has many incomplete properties. Users can be limited by the project with this version.

Table of Contents

Chapter-1: Introduction

Video Game

A video game is an electronic game that involves human interaction with a user interface to generate visual feedback on a video device such as a TV screen or computer monitor. The word *video* in *video game* traditionally referred to a raster display device, but in the 2000s, it implies any type of display device that can produce two- or three-dimensional images. Video games are sometimes believed to be a form of art, but this designation is controversial.

Two girls playing a driving-themed video arcade game in 2007.
The electronic systems used to play video games are known as platforms; examples of these are personal computers and video game consoles. These platforms range from large mainframe computers to small handheld computing devices. Specialized video games such as arcade games, in which the video game components are housed in a large chassis, while common in the 1980s, have gradually declined in use due to the widespread availability of home video game devices (e.g., PlayStation 4 and Xbox One) and video games on desktop and laptop computers and smartphones.
The input device used for games, the game controller, varies across platforms. Common controllers include gamepads, mouses, keyboards, joysticks, the touchscreens of mobile devices and buttons. In addition to video and (in most cases) audio feedback, some games in the 2000s include haptic, vibration or force feedback peripherals.[1]

Image-1

Video Game Development

Video game development is the process of creating a video game. Development is undertaken by a game developer, which may range from one person to a large business. Traditional commercial PC and console games is normally funded by a publisher and take several years to develop. Indie games can take less time and can be produced cheaply by individuals and small developers. The indie game industry

has seen a rise in recent years with the growth of new online distribution systems and the mobile game market.[2]

Library (Computing)

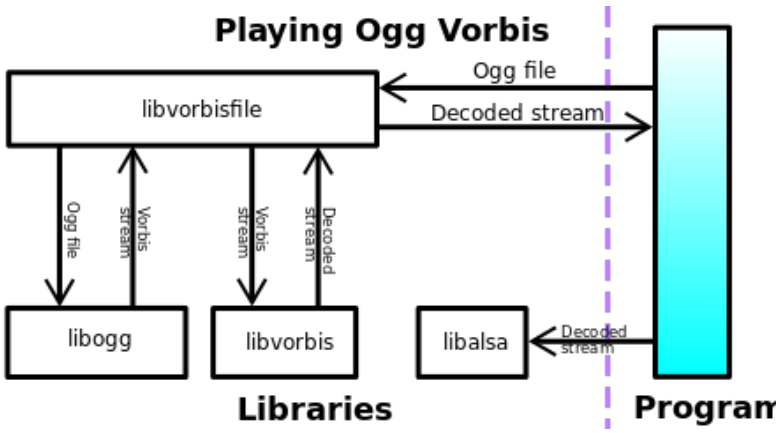
In computer science, a library is a collection of non-volatile resources used by computer programs, often to develop software. These may include configuration data, documentation, help data, message templates, pre-written code and subroutines, classes, values or type specifications. In IBM's OS/360 and its successors they are referred to as partitioned data sets.

In computer science, a library is a collection of implementations of behavior, written in terms of a language, that has a well-defined interface by which the behavior is invoked. This means that as long as a higher level program uses a library to make system calls, it does not need to be re-written to implement those system calls over and over again. In addition, the behavior is provided for reuse by multiple independent programs. A program invokes the library-provided behavior via a mechanism of the language. For example, in a simple imperative language such as C, the behavior in a library is invoked by using C's normal function-call. What distinguishes the call as being to a library, versus being to another function in the same program, is the way that the code is organized in the system.

Library code is organized in such a way that it can be used by multiple programs that have no connection to each other, while code that is part of a program is organized to only be used within that one program. This distinction can gain a hierarchical notion when a program grows large, such as a multi-million-line program. In that case, there may be internal libraries that are reused by independent sub-portions of the large program. The distinguishing feature is that a library is organized for the purposes of being reused by independent programs or sub-programs, and the user only needs to know the interface, and not the internal details of the library.

The value of a library is the reuse of the behavior. When a program invokes a library, it gains the behavior implemented inside that library without having to implement that behavior itself. Libraries encourage the sharing of code in a modular fashion, and ease the distribution of the code.[3]

Image-2



1.1 Motivation

Nowadays, some people all around the world interested in developing video game. However, many of them don't have enough information to develop a game. Therefore libraries help to developers calculating the complex functions, allowing to use easier methods etc.

When a developer develop a complex system. S/he has to learn everything about the topic of the system. If s/he does not know something, different bugs could be happened but if s/he use a library they should not know everything about the system. Because experts of these topics have been written more safely. Therefore, using a library can provide with simplicity to the developer.

1.2 Technology Overview

Java:

Java is a general-purpose computer programming language that is concurrent, class-based, object-oriented,[13] and specifically designed to have as few implementation dependencies as possible. It is intended to let application developers "write once, run anywhere" (WORA),[14] meaning that compiled Java code can run on all platforms that support Java without the need for recompilation.[15] Java applications are typically compiled to bytecode that can run on any Java virtual machine (JVM) regardless of computer architecture. As of 2016, Java is one of the most popular programming languages in use,[16][17][18][19] particularly for client-server web applications, with a reported 9 million developers[4]

The language derives much of its syntax from C and C++, but it has fewer low-level facilities than either of them.

Image-3

WORE:

"Write once, run anywhere" (WORA), or sometimes *write once, run everywhere* (WORE), is a slogan created by Sun Microsystems to illustrate the cross-platform benefits of the Java language.[1][2] Ideally, this means Java can be developed on any device, compiled into a standard bytecode and be expected to run on any device equipped with a Java virtual machine (JVM). The installation of a JVM or Java interpreter on chips, devices or software packages has become an industry standard practice.[5]

Object-Orianted Programming:

Object-oriented programming (OOP) is a programming paradigm based on the concept of "objects", which may contain data, in the form of fields, often known as *attributes;* and code, in the form of procedures, often known as *methods.* A feature of objects is that an object's procedures can access and often modify the data fields of the object with which they are associated (objects have a notion of "this" or "self"). In OOP, computer programs are designed by making them out of objects that interact with one another.[1][2] There is significant diversity of OOP languages, but the most popular ones are class-based, meaning that objects are instances of classes, which typically also determine their type.
Many of the most widely used programming languages are multi-paradigm programming languages that support object-oriented programming to a greater or lesser degree, typically in combination with imperative, procedural programming. Significant object-oriented languages include Common Lisp, Python, C++, Objective-C, Smalltalk, Delphi, Java, Swift, C#, Perl, Ruby, and PHP.[6.1]

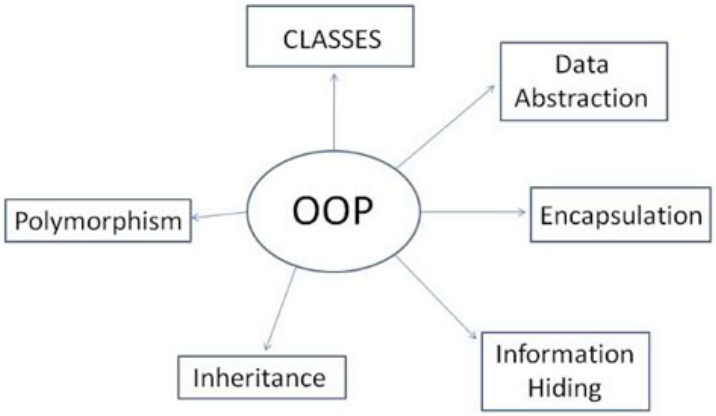


Image-4

Languages that support object-oriented programming typically use inheritance for code reuse and extensibility in the form of either classes or prototypes. Those that use classes support two main concepts:

- Classes - the definitions for the data format and available procedures for a given type or class of object; may also contain data and procedures (known as class methods) themselves
- Objects - instances of classes

Objects sometimes correspond to things found in the real world. For example, a graphics program may have objects such as "circle", "square", "menu". An online shopping system might have objects such as "shopping cart", "customer", and "product".[7] Sometimes objects represent more abstract entities, like an object that represents an open file, or an object which provides the service of translating measurements from U.S. customary to metric.
Each object is said to be an instance of a particular class (for example, an object with its name field set to "Mary" might be an instance of class Employee).
Procedures in object-oriented programming are known as methods; variables are

also known as fields, members, attributes, or properties. This leads to the following terms:

- Class variables - belong to the *class as a whole*; there is only one copy of each one
- Instance variables or attributes - data that belongs to individual *objects*; every object has its own copy of each one
- Member variables - refers to both the class and instance variables that are defined by a particular class
- Class methods - belong to the *class as a whole* and have access only to class variables and inputs from the procedure call
- Instance methods - belong to *individual objects*, and have access to instance variables for the specific object they are called on, inputs, and class variables

Objects are accessed somewhat like variables with complex internal structure, and in many languages are effectively pointers, serving as actual references to a single instance of said object in memory within a heap or stack. They provide a layer of abstraction which can be used to separate internal from external code. External code can use an object by calling a specific instance method with a certain set of input parameters, read an instance variable, or write to an instance variable. Objects are created by calling a special type of method in the class known as a constructor. A program may create many instances of the same class as it runs, which operate independently. This is an easy way for the same procedures to be used on different sets of data.

Object-oriented programming that uses classes is sometimes called class-based programming, while prototype-based programming does not typically use classes. As a result, a significantly different yet analogous terminology is used to define the concepts of *object* and *instance*.

In some languages classes and objects can be composed using other concepts like traits and mixins.[6.2]

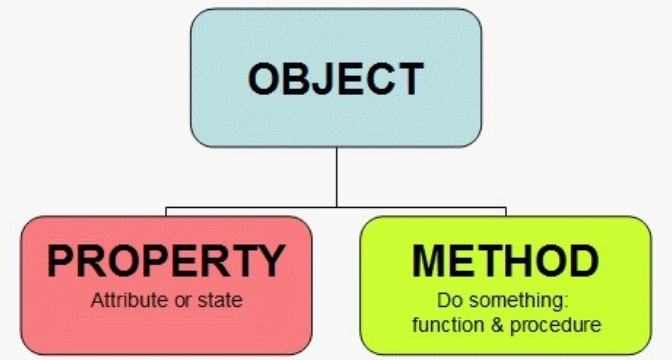
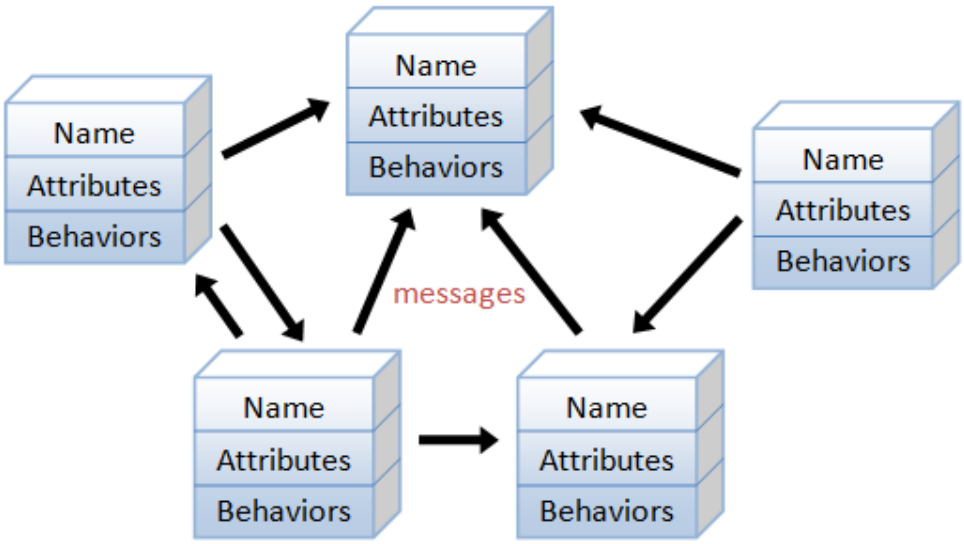


Image-5

Objects can contain other objects in their instance variables; this is known as object composition. For example, an object in the Employee class might contain (point to) an object in the Address class, in addition to its own instance variables like "first_name" and "position". Object composition is used to represent "has-a" relationships: every employee has an address, so every Employee object has a place to store an Address object.[6.3]

Subtyping, a form of polymorphism, is when calling code can be agnostic as to whether an object belongs to a parent class or one of its descendants. For example, a function might call "make_full_name()" on an object, which will work whether the object is of class Person or class Employee. This is another type of abstraction which simplifies code external to the class hierarchy and enables strong separatioof concerns.



An object-oriented program consists of many well-encapsulated objects and interacting with each other by sending messages

Image-6

Chapter-2: System Design

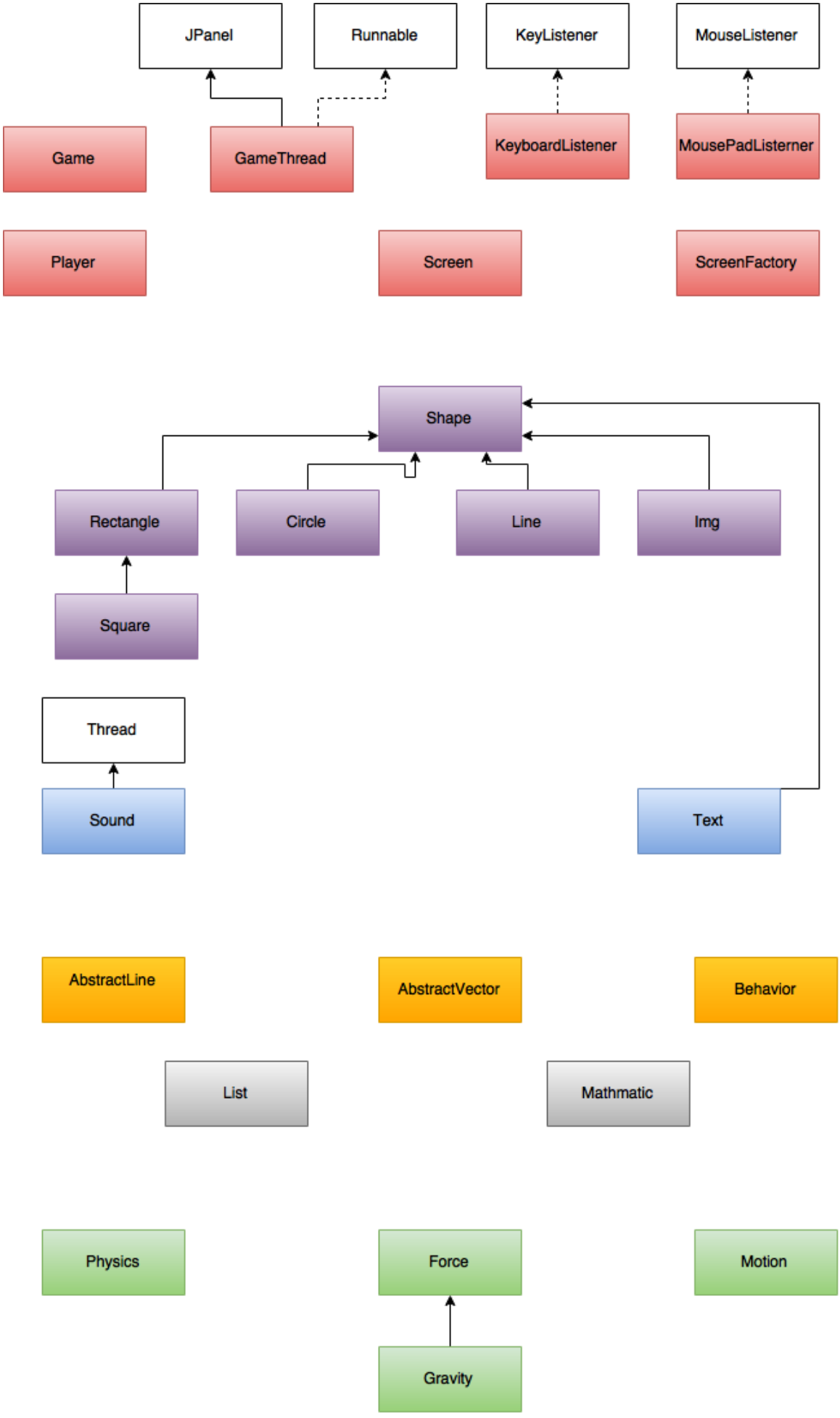
Program was designed by Bilgehan NAL and all the coding was done in Java.

Object-Orianted Design of the Project:

Alpha version of Niyazi Project occurs 24 basic classes and 6 packages.

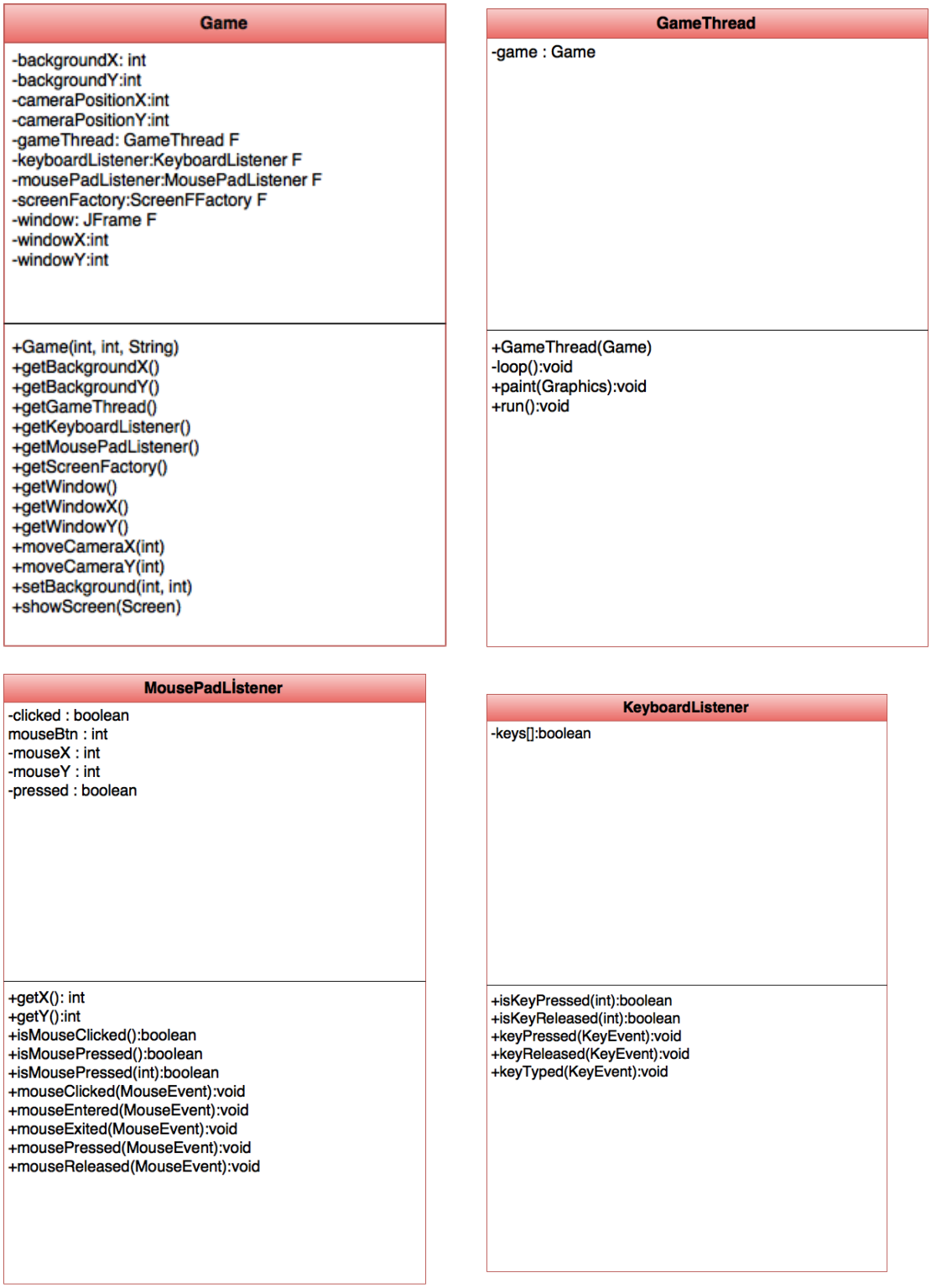
These are:

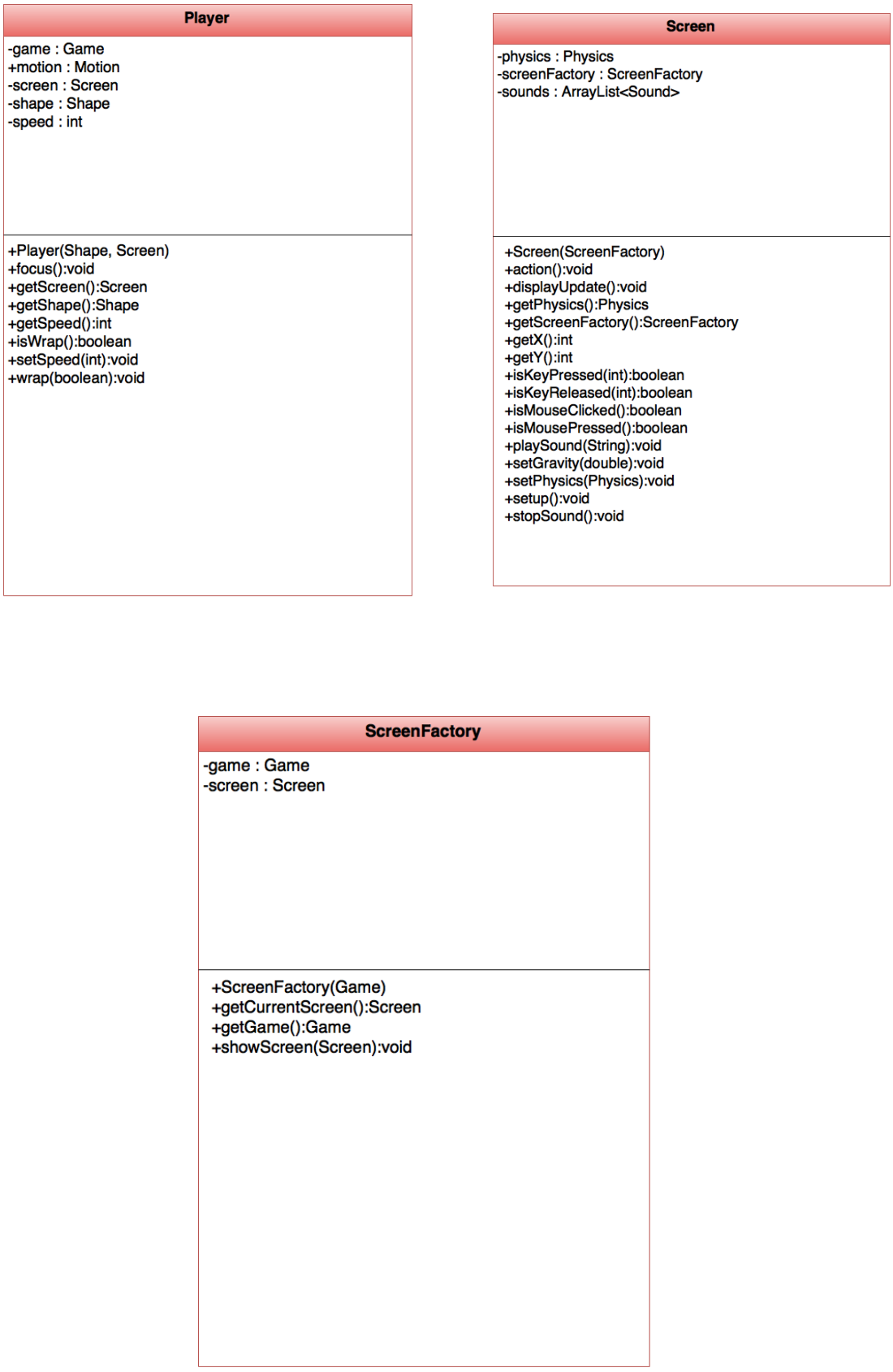
- Com.Substructure
  - Game
  - GameThread
  - KeyboardListener
  - MousePadListener
  - Player
  - Screen
  - ScreenFactory
- Com.Shape
  - Shape
  - Rectangle
  - Circle
  - Line
  - Img
  - Square
- Com.InformationTools
  - Sound
  - Text
- Com.PropertiesOfObjects
  - AbstractLine
  - AbstractVector
  - Behavior
- Com.UsefulItems
  - List
  - Mathmatic
- Com.Physics
  - Physics
  - Force
  - Motion
  - Gravity


JPanel
Runnable
KeyListener
MouseListener
Game
GameThread
KeyboardListener
MousePadListerner
Player
Screen
ScreenFactory
Shape
Rectangle
Circle
Line
Img
Square
Thread
Sound
Text
AbstractLine
AbstractVector
Behavior
List
Mathmatic
Physics
Force
Motion
Gravity

2.1 UML Of Classes

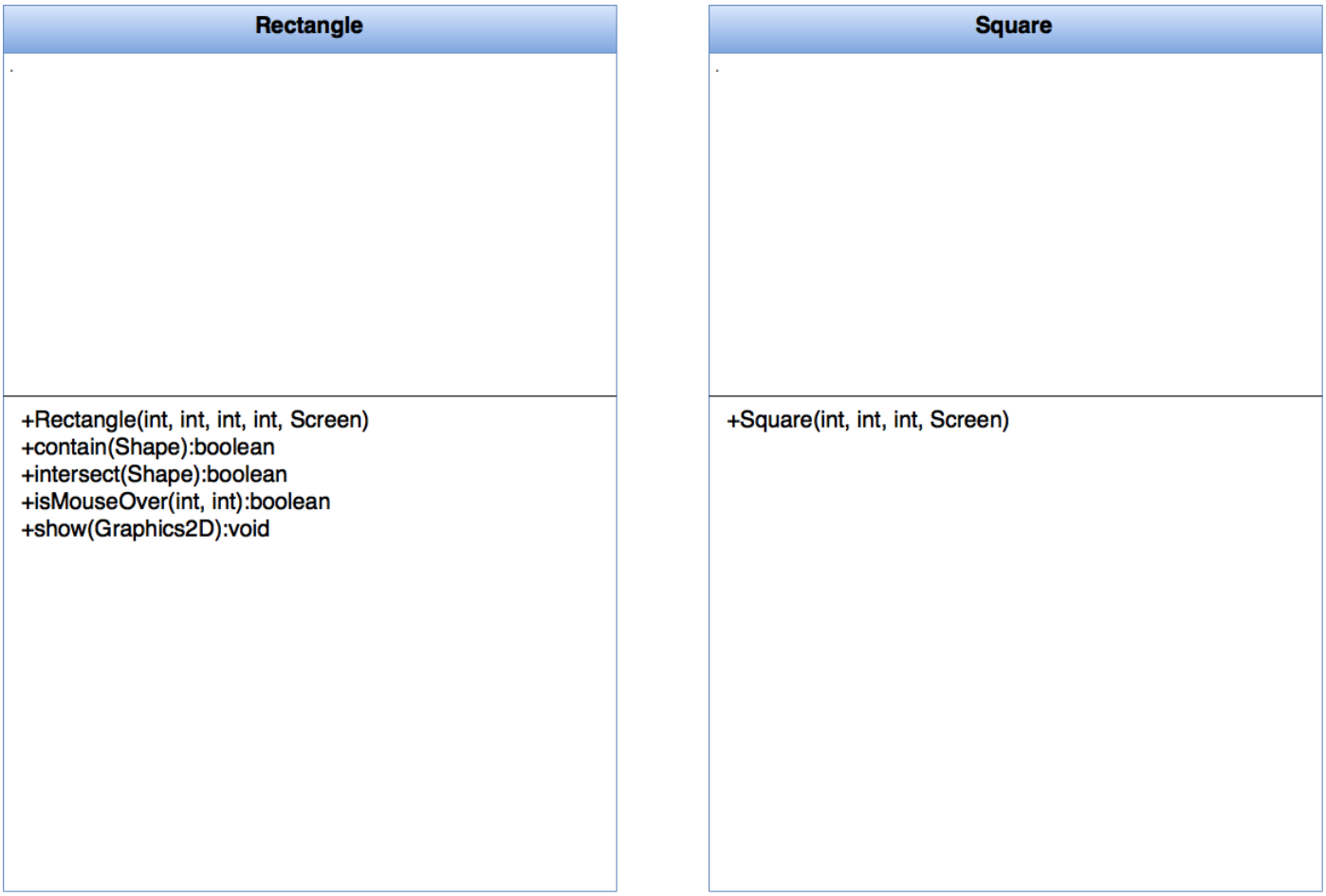
A > com.Substructure



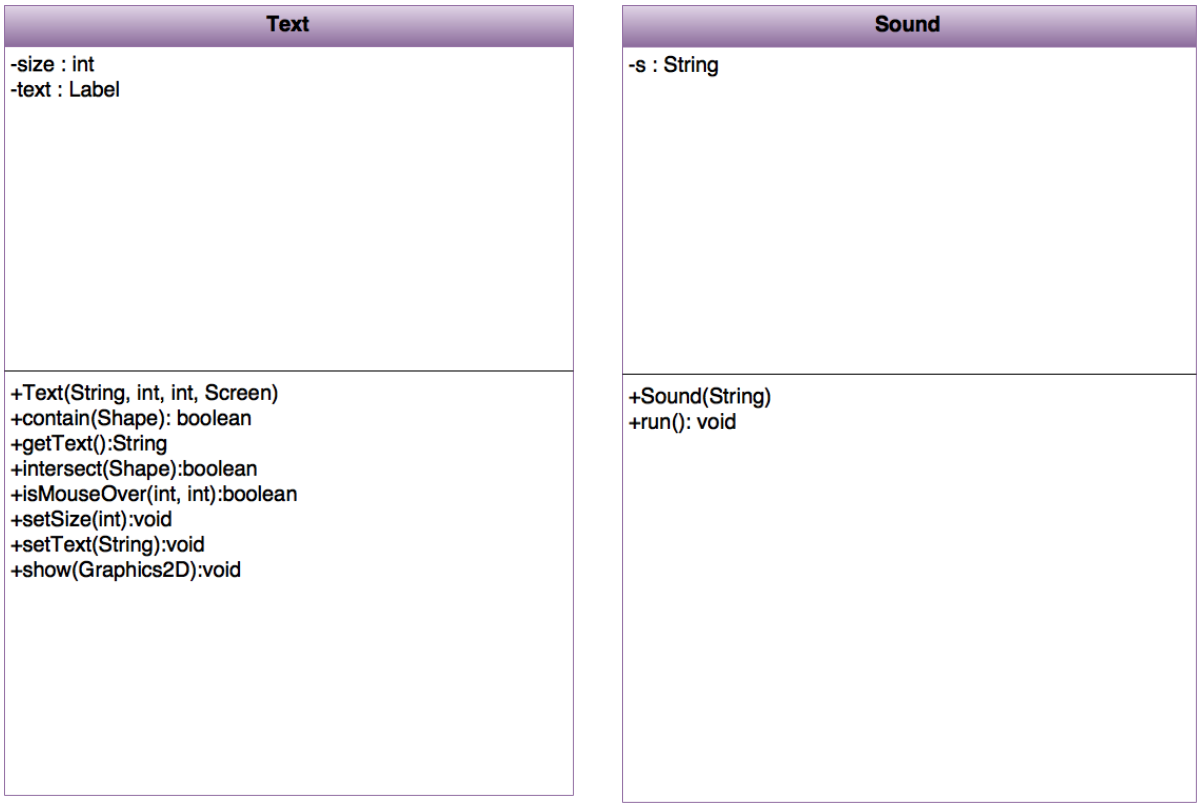

Player
-game : Game
+motion : Motion
-screen : Screen
-shape : Shape
-speed : int
+Player(Shape, Screen)
+focus():void
+getScreen():Screen
+getShape():Shape
+getSpeed():int
+isWrap():boolean
+setSpeed(int):void
+wrap(boolean):void
Screen
-physics : Physics
-screenFactory : ScreenFactory
-sounds : ArrayList<Sound>
+Screen(ScreenFactory)
+action():void
+displayUpdate():void
+getPhysics():Physics
+getScreenFactory():ScreenFactory
+getX():int
+getY():int
+isKeyPressed(int):boolean
+isKeyReleased(int):boolean
+isMouseClicked():boolean
+isMousePressed():boolean
+playSound(String):void
+setGravity(double):void
+setPhysics(Physics):void
+setup():void
+stopSound():void
ScreenFactory
-game : Game
-screen : Screen
+ScreenFactory(Game)
+getCurrentScreen():Screen
+getGame():Game
+showScreen(Screen):void

B > com.Shape

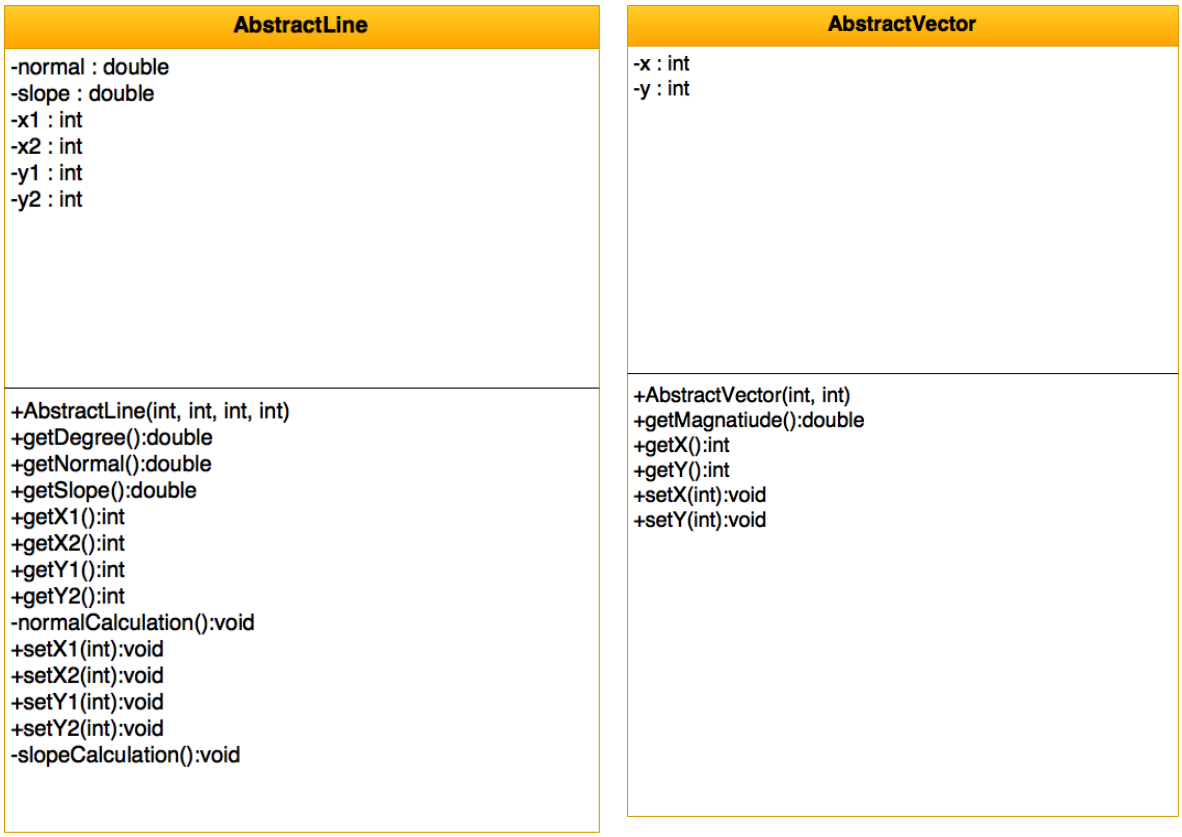

Shape
-backGroundX : int
-backgroundY : int
-shapeList : List
-color : Color
-crash : boolean
-crashingLine : AbstractLine
-filled : boolean
-gravity : Gravity
-height : int
-impulsePercentage : double
-intersectingShapes : ArrayList<Shape>
-netForce : Force
-oldPositionX : int
-oldPositionY : int
-positionX : int
-positionY : int
-screen : Screen
-speedLimitX : int
-speedLimitY : int
-speedVector : AbstractVector
-speedX : double
-speedY : double
-velocityDirectionLine : AbstractLine
-visibility : boolean
-weight : double
-width : int
+behaviour:Behaviour
+Shape(int, int, Screen)
+absoluteMotionX(int):void
+absoluteMotionY(int):void
+contain(Shape):boolean
+crash(Shape):boolean
+getBehaviour():Behaviour
+getColor():Color
+getCrashingLine():AbstractLine
+getFilled():boolean
+getGravity():Gravity
+getHeight():int
+getImpulsePercentage():double
+getIntersectingShapes():ArrayLİst<Shape>
+getNetForce():Force
+getOldPositionX():int
+getOldPositionY():int
+getPositionX():int
+getPositionY():int
+getSpeedLimitX():int
+getSpeedLimitY():int
+getSpeedVector():AbstractVector
+getSpeedX():double
+getSpeedY():double
+getVelocityDirectionLine():AbstractLine
+getWeight():double
+getWidth():int
+hide():void
+intersect(Shape):boolean
+isMouseOver(int, int):boolean
+isVisibility():boolean
+setColor(Color):void
+setCrashingLine(AbstractLine):void
+setFilled(boolean):void
+setGravity(Gravity):void
+setHeight(int):void
+setImpulsePercentage(double):void
+setNetForce(Force):void
+setOldPositionX(int):void
+setOldPositionY(int):void
+setPositionX(int):void
+setPositionY(int):void
+setSpeedLimitX(int):void
+setSpeedLimitY(int):void
+setSpeedX(double):void
+setSpeedY(double):void
+setVelocityDirectionLine(AbstractLine):void
+setWeight(double):void
+setWidth(int):void
+show():void
+show(Graphics2D):void

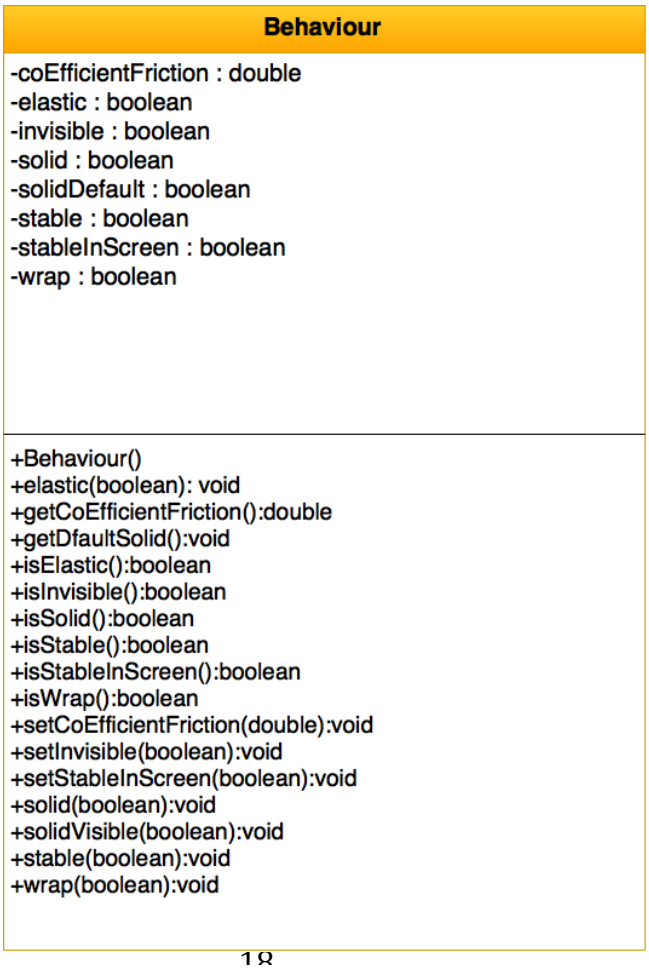

Rectangle
+Rectangle(int, int, int, int, Screen)
+contain(Shape):boolean
+intersect(Shape):boolean
+isMouseOver(int, int):boolean
+show(Graphics2D):void
Square
+Square(int, int, int, Screen)
Img
-icon : ImageIcon
-img : Image
-url : String
+Img(String, int, int, Screen)
+contain(Shape):boolean
+intersect(Shape):boolean
+isMouseOver(int, int):boolean
+show(Graphics2D):void

C > com.InformationTools

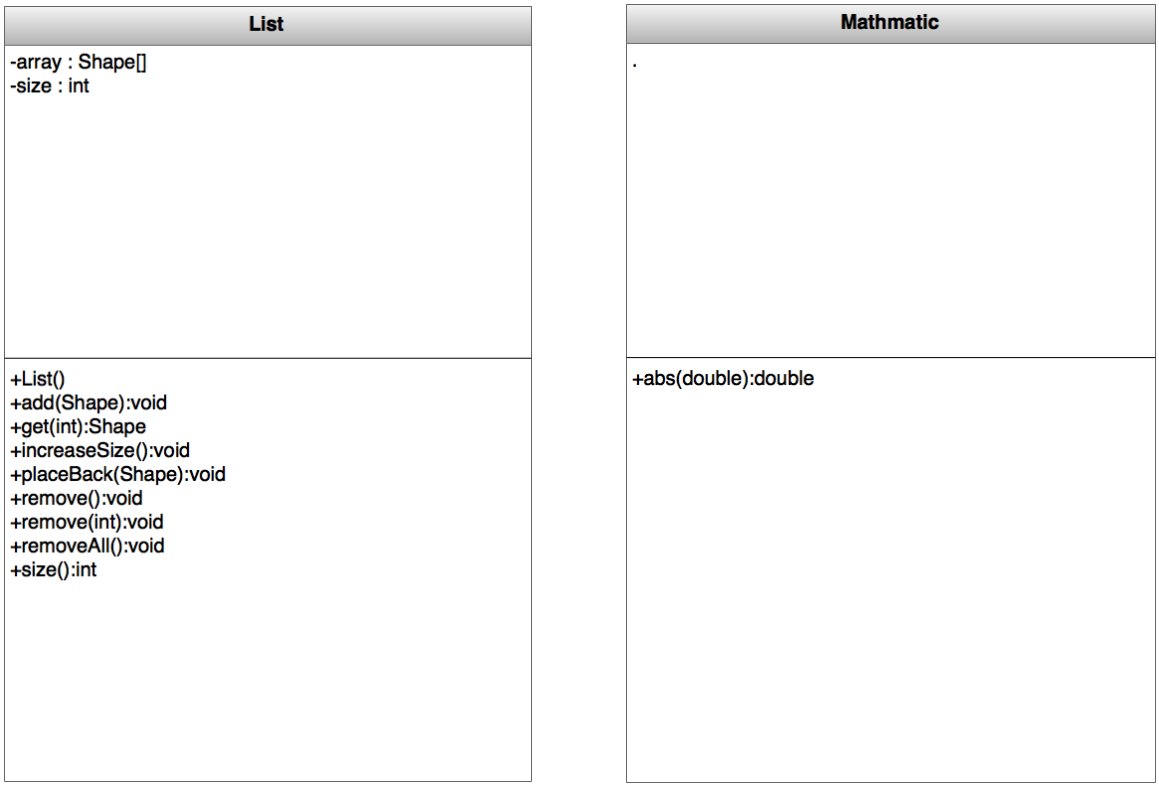


D > com.PropertiesOfObjects

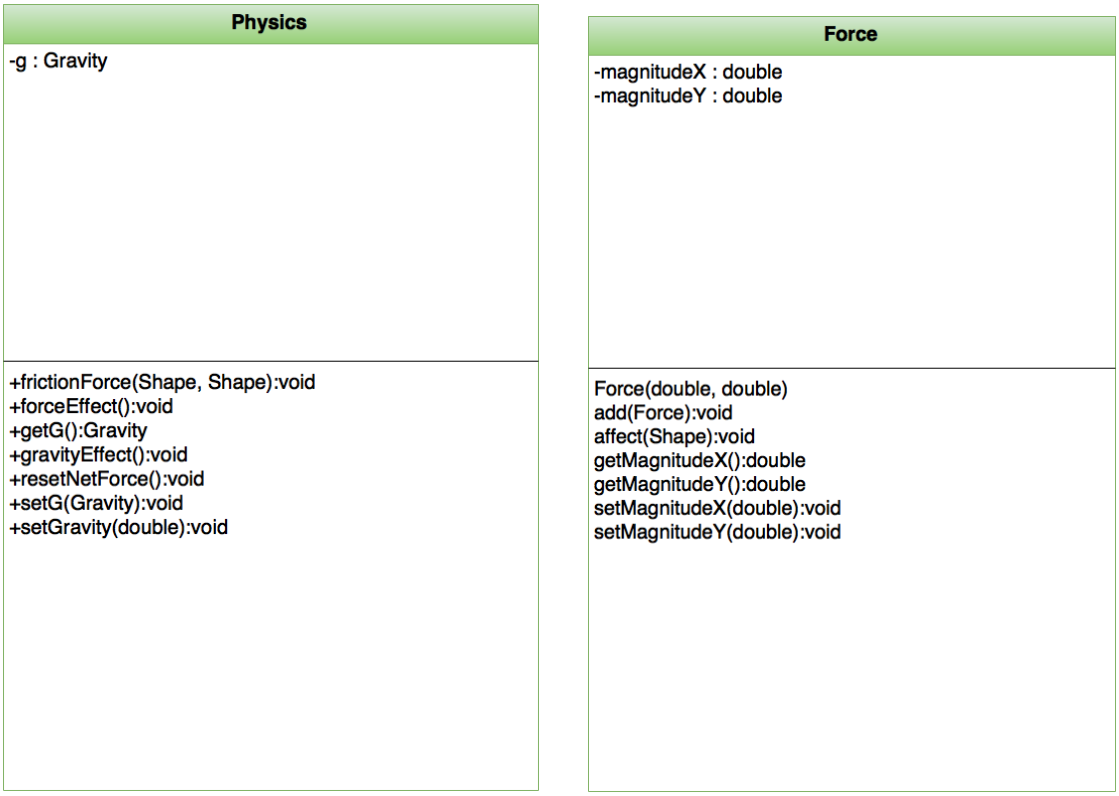


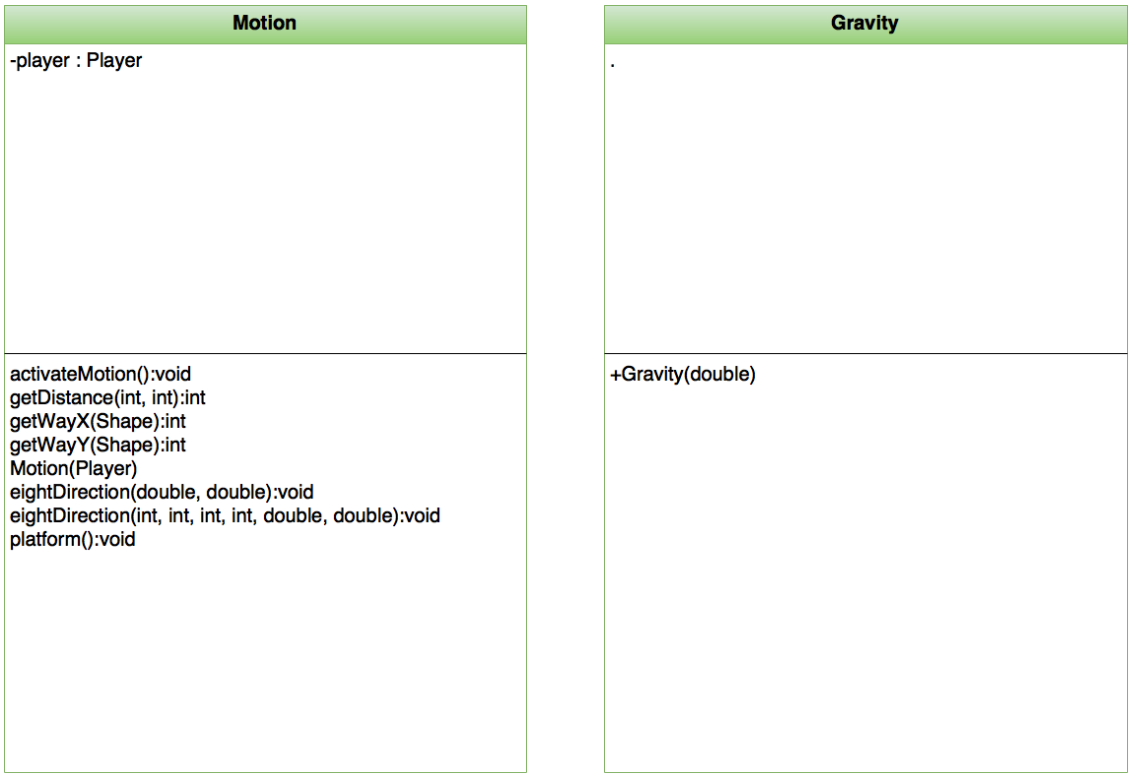


E > com.UsefulItems



F > com.Physics





2.2 Operation of the Project

Project Niyazi was designed as a library. A person who wants to use Project Nİyazi in his/her project, s/he has to import jar file of the library. Then the library is ready to use.

A game which is running with Project Niyazi has minimum a few class and minimum one class has to extend “Screen” class. “Screen” class is a kind of abstract class which has some abstract methods. The IDE which the developer’s prefer recommend to override two methods.

First one of the overriding methods is setup. This method is run when the game is opened just one time. Therefore the developer should write settings of his/her project in this method or s/he should write their code which is run one time.

Second one of the overriding methods is action method. In this method, a while loop runs. Therefore this method provides with the running of the program continuously. All changes which is made by the user operate in this class every 12 millisecond(average).

Project Niyazi provide with huge eases to the developers. It has many method to create object easily and to compute complex physicial behaviors of the objects. Only thing that user requires to do is using the available methods of the library.

Finally Niyazi project uses an external library to play sounds.

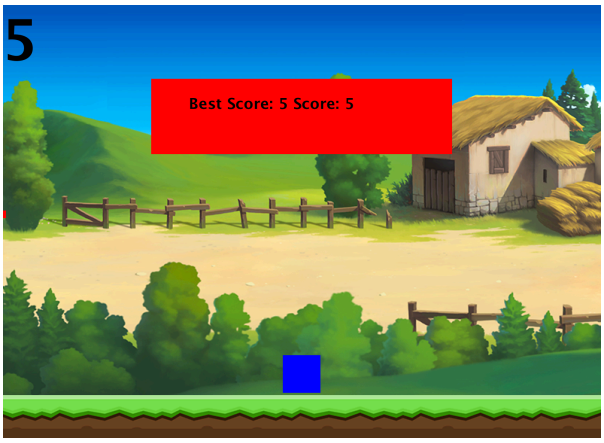
3.1 Future Work

- More shapes will be available to use in the game.
- Collusion detection will be improved and its architecture will be redesigned.
- Rotational motion properties will be added to the project.
- Calculations of the objects which is stand without 0 or 90 degree will be calculated more efficiently using anlatic geometry.
- Multi window games will be supported better.

Chapter-4: Test

In this project, a very simple 2d game was created. The test game has all physicial properties and all motions are calculating according to physicial formulas.

Some pictures from the test game:



References:

[1]: https://en.wikipedia.org/wiki/Video_game

[2]: https://en.wikipedia.org/wiki/Video_game_development

[3]: https://en.wikipedia.org/wiki/Library_(computing)

[4]: https://en.wikipedia.org/wiki/Library_(computing)

[5]: https://en.wikipedia.org/wiki/Library_(computing)

[6]: https://en.wikipedia.org/wiki/Object-oriented_programming

Image[1]: https://en.wikipedia.org/wiki/Object-oriented_programming

Image[2]: https://en.wikipedia.org/wiki/Library_(computing)#/media/File:Ogg_vorbis_libs_and_application_dia.svg

Image[3]: https://en.wikipedia.org/wiki/Library_(computing)#/media/File:Ogg_vorbis_libs_and_application_dia.svg

Image[4]: http://www.selcuk-aydin.com/wp-content/uploads/2014/05/oops.jpg

Image[5]: http://img.wonderhowto.com/img/89/23/63557958543427/0/hack-like-pro-python-scripting-for-aspiring-hacker-part-2.w654.jpg

Image[6]: https://www3.ntu.edu.sg/home/ehchua/programming/java/images/OOP_Objects.png