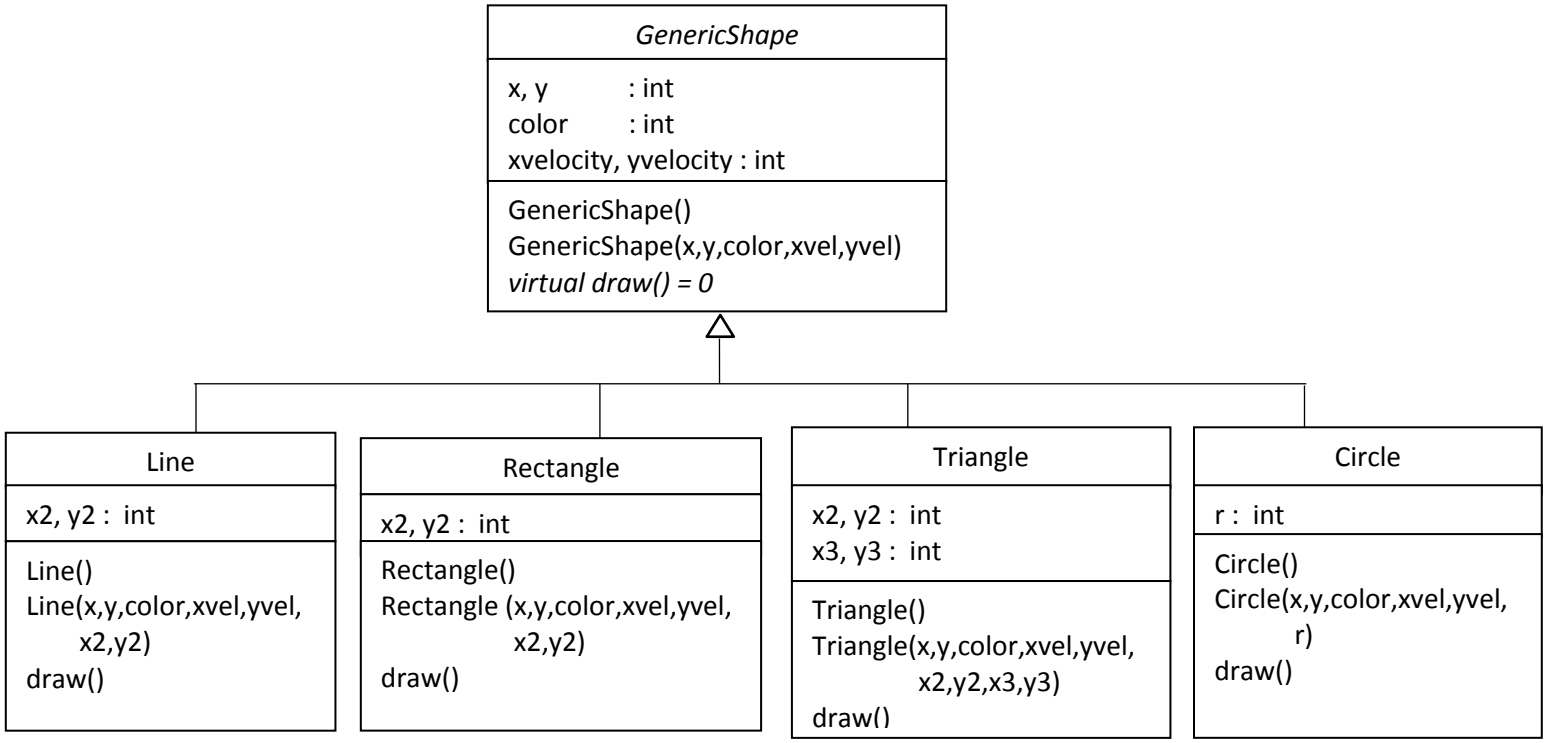
BLG252E - Object Oriented Programming
HOMEWORK #2

Start : 07.11.2013
Due Date: 28.11.2013
(3 Weeks)

In this homework, you will write a C++ program for **bouncing animation** of two-dimensional graphical objects.

You should define an abstract base class (GenericShape) , which is similar to the example (e84a.cpp) described in the lecture. The **draw()** function of the **base** class should be defined as **pure virtual** function, so that it can be implemented as **polymorphic** functions in derived classes.

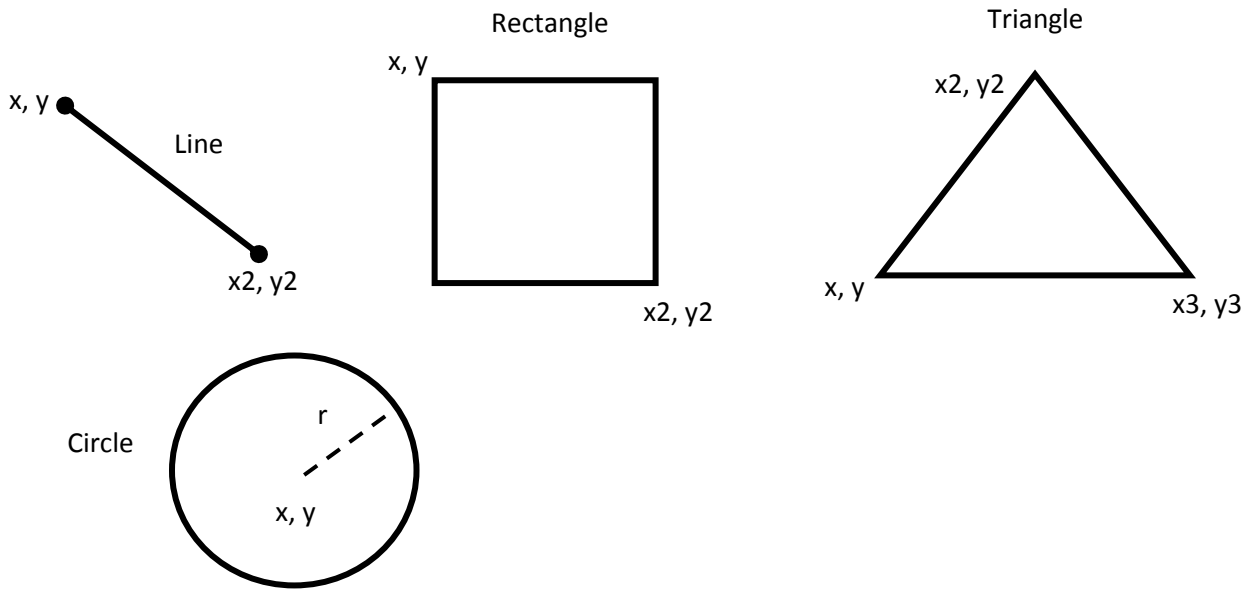
The following class diagram shows the derivation hierarchy.



For each class, you should write two constructor functions:

1) Default constructor (without any input parameters)
2) Parameterized constructor

ATTRIBUTES OF SHAPES



GRAPHICS LIBRARY

In your program, you will use the BGI graphics library (Borland Graphics Interface) for Windows.

The BGI source files (**winbgi2.cpp** and **graphics2.h**) , and also the complete BGI reference documentation (index.htm) is provided to you as part of this homework.

The following is an example program (non-object-oriented), which can work in any compiler such as Dev-C++ 5.0 , or Microsoft Visual Studio 2012.

```
// Ornek.cpp

#include <stdio.h>
#include <stdlib.h>
#include "graphics2.h"

int main()
{
    int GraphDriver=0,GraphMode=0;
    initgraph( &GraphDriver, &GraphMode, "", 640, 480 );  // Start graphics window

    printf("This is the console window ... \n");
    outtext("This is the graphics window ... ");

    setcolor(RED);
    line(50,150, 180, 60);  // Draws Line
```

```
    setfillstyle(SOLID_FILL, RED);
    bar(150,200, 250, 260);   // Rectangle

    setfillstyle(SOLID_FILL, BLUE);
    fillellipse (400,200,50,50);   // Circle

    // The vertex pairs (x,y) of a triangle should be assigned to a one-dimensional polygon array.
    // The last element pair and the first element pair should be the same, so that the polygon is closed.
    int poly[8] = {100,150,200,150,175,80,100,150};
    setfillstyle(SOLID_FILL, GREEN);
    fillpoly(4, poly);  // Triangle

    outtext ("Animation has finished, press a key.");
    system("pause");
    return 0;
}
```

The followings are the BGI library function prototypes used in the example program.

```
void initgraph (int *graphdriver, int *graphmode, char *pathtodriver);
void outtext (char *textstring);
void setcolor (int color);
void line (int x1, int y1, int x2, int y2);
void setfillstyle (int pattern, int color);
void bar (int left, int top, int right, int bottom);
void fillellipse (int x, int y, int xradius, int yradius);
void fillpoly (int numpoints, int *polypoints);
```

(For all other library functions, see the **graphics2.h** file, or the **index.htm** file of BGI documentation.)

Compiling the example program with in Dev-C++

1) Add the following three files into your project file: **ornek.cpp, winbgi2.cpp, graphics2.h**

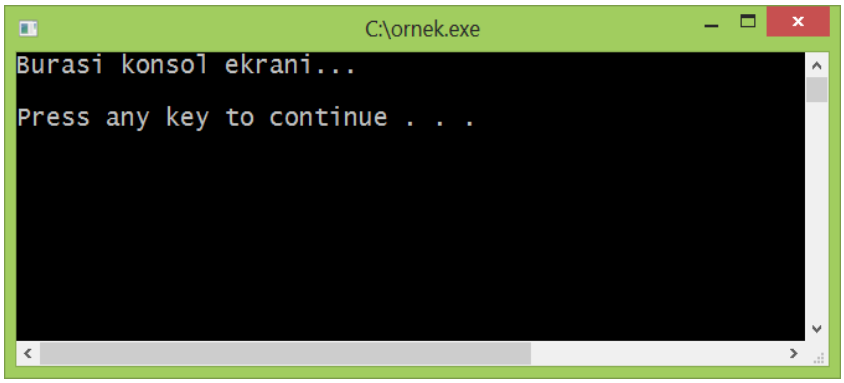
2) Set the following linker switch option.
- Go to "Project" menu and choose "Project Options".
- Go to "Parameters" tab.
- In "Linker" field, enter "**-lgdi32**"

3) Compile and run program.

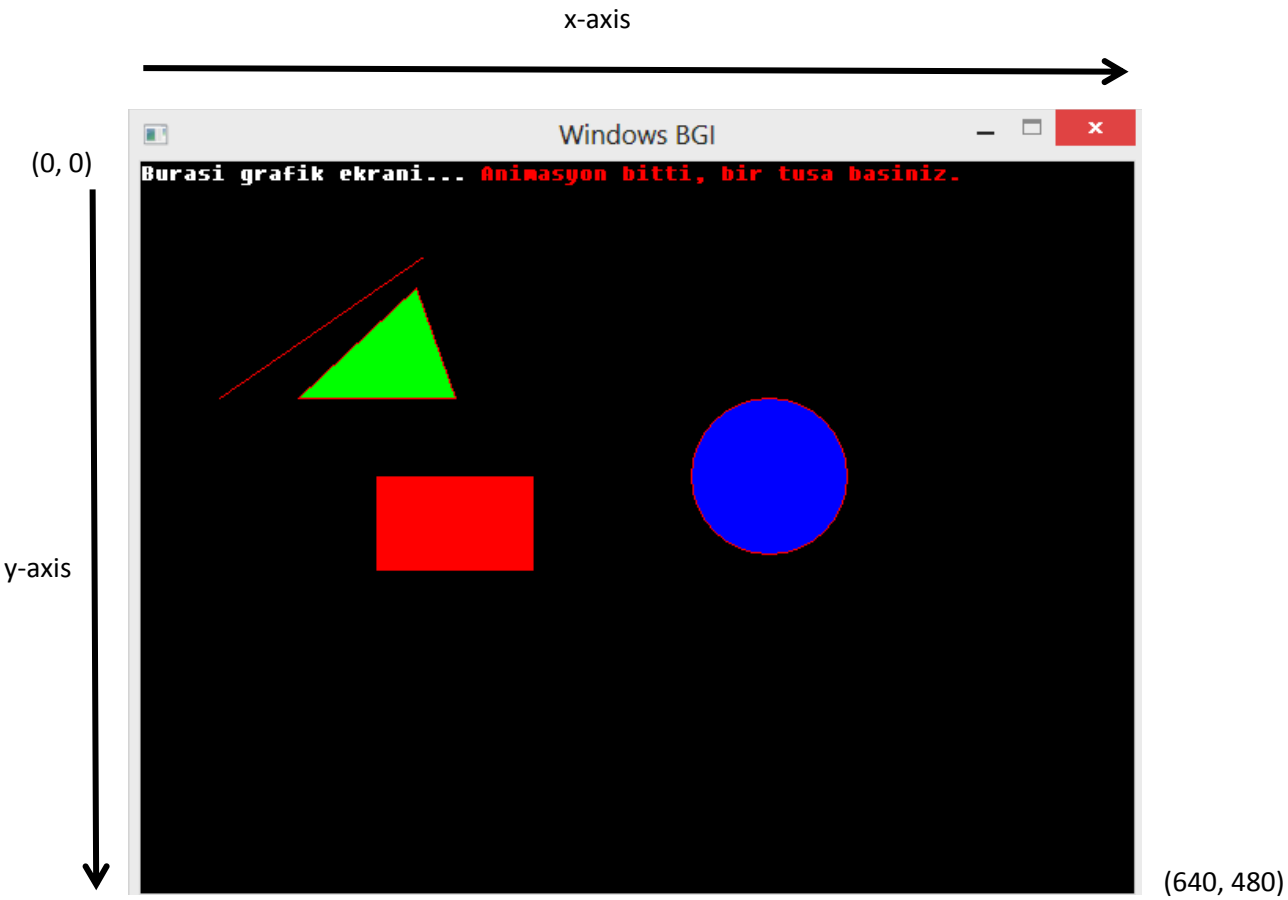
OUTPUT SCREENS

When you run the example program, two windows are created:

Console window : For standard I/O functions such as scanf(), printf(), cin, cout.



Graphics window: For BGI library functions, such as initgraph(), line(),outtext(),setcolor(), etc., and also mouse inputs.



ANIMATION ALGORITHM

PHASE #1)

In the main() program, define the following 4 objects, by calling the parameterized constructors, with the given input parameter data.

Object #	Object Type	Constructor Input Parameters									
		color	xvel	yvel	x	y	x2	y2	x3	y3	r
1	Line	4	2	-3	50	150	180	60			
2	Rectangle	4	5	4	150	200	250	260			
3	Circle	1	-3	6	400	200					50
4	Triangle	2	4	4	100	150	200	150	175	80	

After creating the 4 objects described above, the main() program should start a while loop until the user hits a key on the keyboard:

```
while (! kbhit( ) )  { …… }
```

Inside the loop, perform the followings:

1. Draw all 4 objects one by one with their specified colors.
2. Wait for a certain amount of time (you may write an empty loop).
3. Redraw all objects one by one again, this time with the background color (default BLACK) so that they will be invisible.
4. Calculate new coordinate positions for all objects. For this purpose, the xvel and yvel values (velocities) should be added to the current x,y, x1,y1, x2,y2, x3,y3 values.
5. Check if the new coordinate positions are within the range of the graphics window borders ($0 < x < 640$, and $0 < y < 480$).
   If not, reverse the xvel and the yvel values. This is done easily by multiplying the xvel by -1, or by multiplying the yvel by -1. This operation will cause the object bounce from the window borders.

PHASE #2)

In this phase, define the following 4 other objects, by calling the default constructors, <u>without</u> giving any input parameter data.

Object #	Object Type	Constructor Values									
		color	xvel	yvel	x	y	x2	y2	x3	y3	r
5	Line	Random	Random	Random	320	240	Random	Random			
6	Rectangle	Random	Random	Random	320	240	Random	Random			
7	Circle	Random	Random	Random	320	240					Random
8	Triangle	Random	Random	Random	320	240	Random	Random	Random	Random	

The **<u>default constructor</u>** of the **<u>base class</u>** should assign the followings:

- x = 320 and y = 240 (Fixed values)
- color = Random number between 1 and 15
- xvel and yvel = Random numbers between -10 and 10

The **default constructors** of the **derived classes** should assign their related variables with the followings:

- x2 = x ± Random number between 50 and 100
- y2 = y ± Random number between 50 and 100
- x3 = x ± Random number between 50 and 100
- y3 = y ± Random number between 50 and 100
- r = Random number between 20 and 80

You should use the following method to generate a random integer number between **a** and **b**.

```
srand( time(NULL) );        // Seed random number generator

number = a + ( rand() % (b-a+1) );
```

The built-in functions srand() and rand() are included in **<stdlib.h>** header file.

After creating the 4 objects described above, the main() program should repeat the same while loop that you have implemented in Phase1.