# ASSIGNMENT

# JAVA AND GRAPHICS

# GRAPHICS

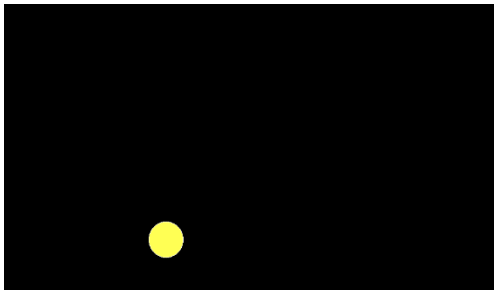
1. Program to show Bouncing Ball:

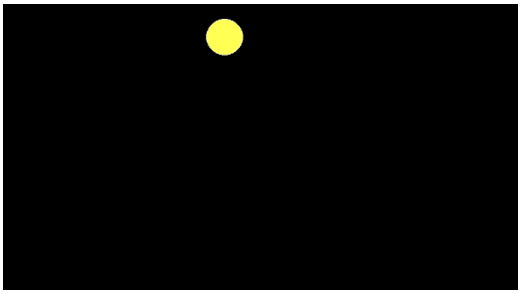
## Source Code:

```
#include<stdio.h>
#include<conio.h>
 #include<graphics.h>
int main(){
int gd=DETECT,gm,x,y;
initgraph(&gd,&gm,"c:\\turboc3\\bgi");
setfillstyle(SOLID_FILL,YELLOW);
for(x=50;x<600;x++)
{
for(y=30;y<455;y+=20)
{
x+=5;
cleardevice();
circle(x,y,30);
floodfill(x,y,WHITE);
delay(60);
}
for(y=435;y>30;y-=20)
{
x+=5;
cleardevice();
circle(x,y,30);
floodfill(x,y,WHITE);
delay(60);
```

```
}
}
getch();
closegraph();
return 0;
}
```

Output:





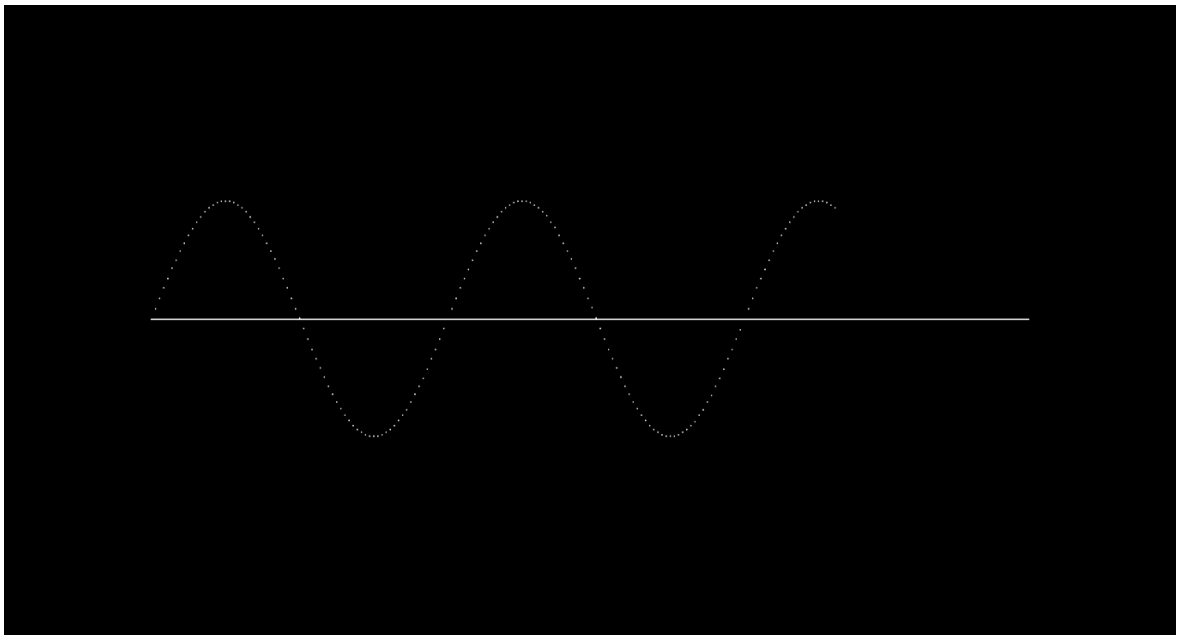
## 2. SIN GRAPH:

### Source Code:

```
#include<stdio.h>
#include<conio.h>
#include<graphics.h>
#include<math.h>
void main() {
int gd=DETECT,gm;
int angle;
int amplitude;
double x,y;
printf("enter the angle:");
```

```
scanf("%d",&angle);
printf("enter the amplitude");
scanf("%d",&amplitude);
initgraph(&gd,&gm,"c:\\turboc3\\bgi");

line(0,getmaxy()/2,getmaxx(),getmaxy()/2);
for(x=0;x<500;x=x+3){
 y=amplitude*sin(angle*3.141/180);
 y=getmaxy()/2-y;
 putpixel(x,y,15);
 delay(100);
 angle=angle+5;
 }
 getch();
 closegraph();
 }
```

Output:



3. Program to draw Concentric Circle:

Source Code:

```
#include<graphics.h>
#include<stdio.h>
#include<conio.h>
int main()
{
int gd=DETECT,gm;
int x,y,i,r=30;;
initgraph(&gd,&gm,"C:\\TC\\bgi");
x=getmaxx()/2;
y=getmaxy()/2;
outtextxy(240,50,"concenteric circle");
for(i=0;i<6;i++){
```

```
setcolor(i);
circle(x,y,r);
r+=10;
}
getch();
closegraph();
return 0;
}
```

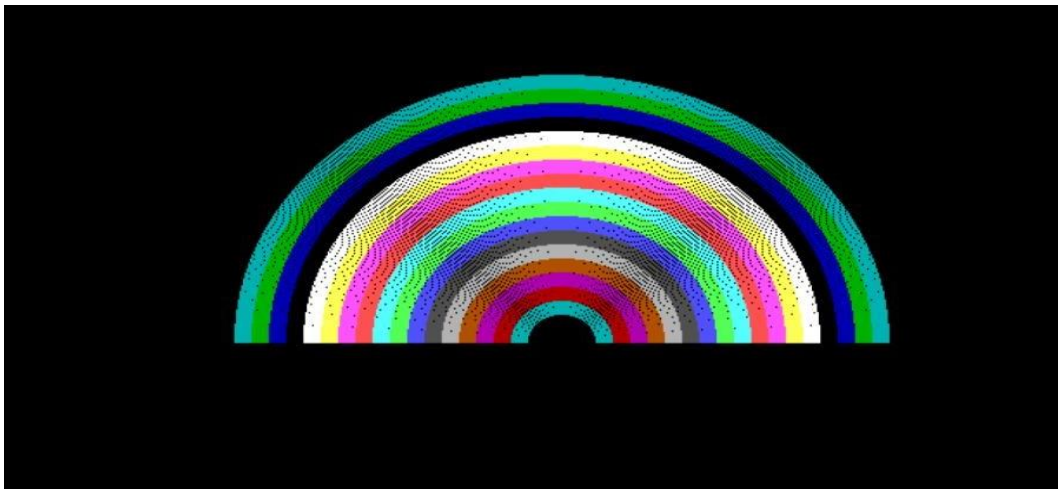
OUTPUT:


4. Program to draw rainbow.

Source Code:

```
#include<stdio.h>
#include<conio.h>
#include<graphics.h>
#include<dos.h>
void main()
{
int gdriver =
DETECT,gmode;
int x,y,i;
initgraph(&gdriver,&gmo
de,"C:\\Turboc3\\BGI");
x=getmaxx()/2;
y=getmaxy()/2;
for(i=30;i<200;i++)
{
delay(100);
setcolor(i/10);
arc(x,y,0,180,i-10);
}
getch();
}
```

## OUTPUT:



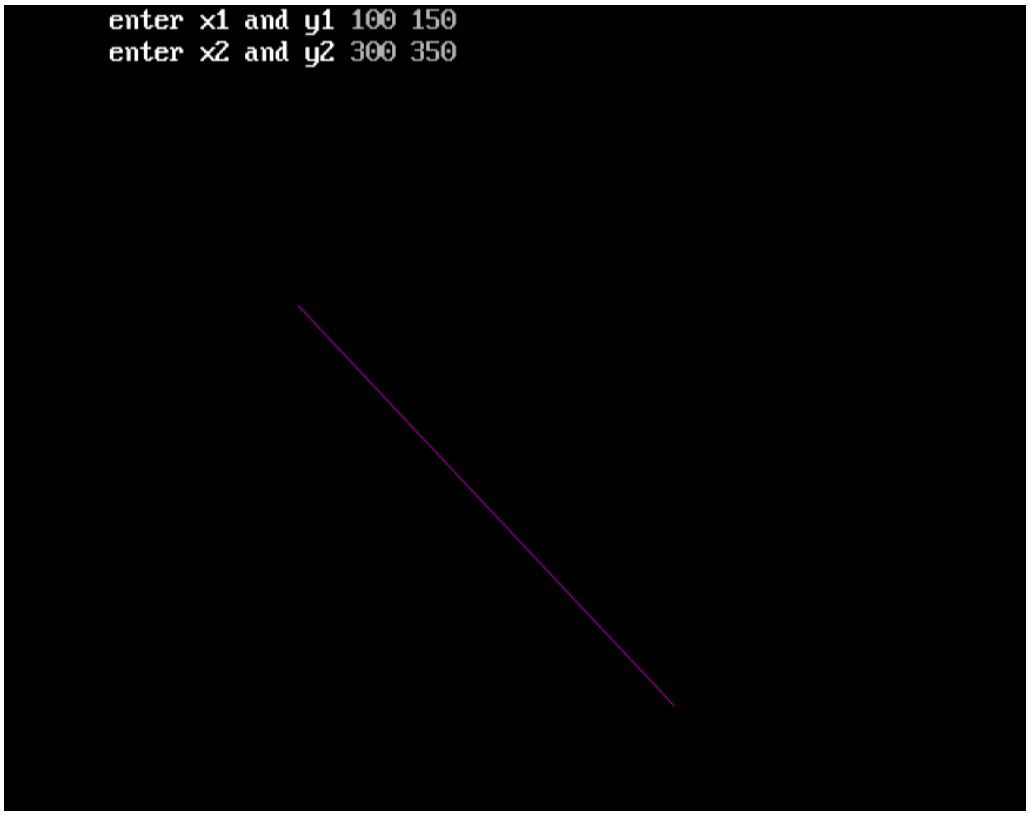
## 5. DDA line Algorithm:

### Source Code:

```
#include<stdio.h>
#include<graphics.h>
#include<math.h>
#include<dos.h>
#include<conio.h>

void main()
{
//clrscr();
int i,gd=DETECT,gm;
float x,y,x1,x2,y1,y2,dx,dy,step;
initgraph(&gd,&gm,"C\\TURBOC3\\BGI");
//float x,y,x1,x2,y1,y2,dx,dy,step;
printf("enter x1 and y1 ");
scanf("%f%f",&x1,&y1);
printf("enter x2 and y2 ");
scanf("%f%f",&x2,&y2);
dx=abs(x2-x1);
dy=abs(y2-y1);
```

```
if(dx>=dy)
step=dx;
else
step =dy;
dx=dx/step;
dy=dy/step;
x=x1;
y=y1;
i=1;
 while(i<=step)
{
putpixel(x,y,5);
x=x+dx;
y=y+dy;
i=i+1;
}
getch();
closegraph();
}
```

## OUTPUT:




6. Program to implement Bresenham's line drawing algorithm.

<u>Code:</u>

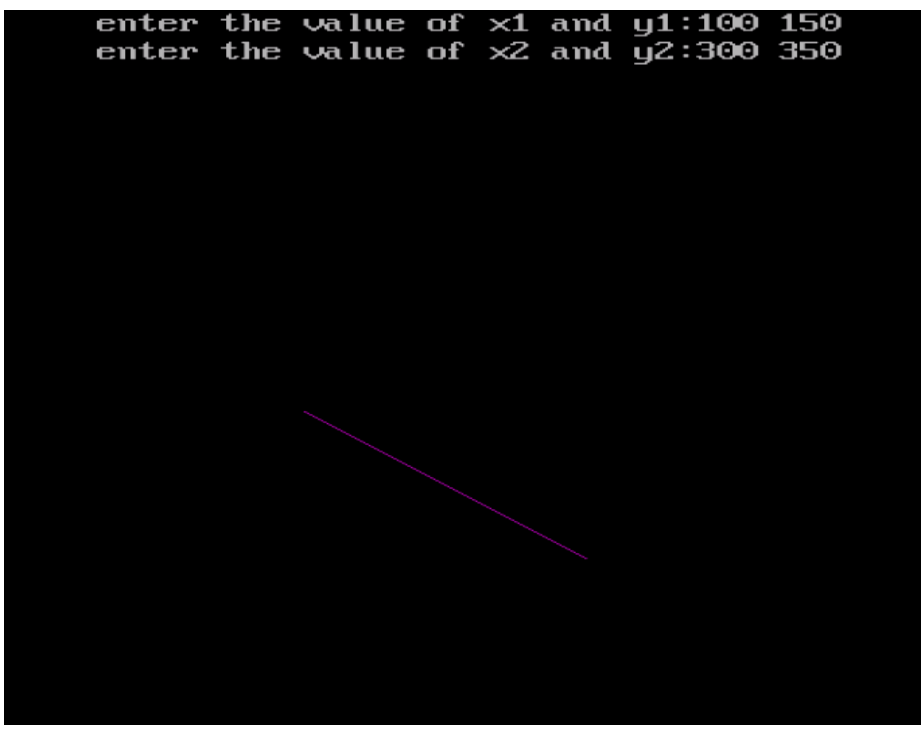
```
#include<graphics.h>
#include<conio.h>
#include<stdio.h>
#include<math.h>
#include<dos.h>
void main(){
int gd=DETECT,gm,i;
```

```
int p,m, x,y,x1,y1,x2,y2,dx,dy,xend;
float slope;
printf("enter the value of x1 and y1:");
scanf("%d%d",&x1,&y1);
printf("enter the value of x2 and y2:");
scanf("%d%d",&x2,&y2);
dx=abs(x2-x1);
dy=abs(y2-y1);
slope=(float)dy/dx;
initgraph(&gd,&gm,"c:\\turboc3\\bgi");
p=(2*dy)-dx;
if(0<slope<1){
if(x1>x2)
{
x=x2;
y=y2;
xend=x1;
}
else{
x=x1;y=y1;
xend=x2;
}
putpixel(x,y,5);
while(x<xend){
if(p<0)
{
x++;
```

```
p+=2*(dy);
}
else{
x++;
y++;
p+=2*(dy-dx);
}
putpixel(x,y,5);
}
}
else{
 if(x1>x2)
{
x=x2;
y=y2;
xend=x1;
}
else{
x=x1;y=y1;
xend=x2;
}
putpixel(x,y,5);
while(x<xend){
if(p<0)
{
x--;
p+=2*(dy);
```

```
}
else{
x--;
y--;
p+=2*(dy-dx);
}
putpixel(x,y,5);
}
getch();
closegraph();
}
}
```

## OUTPUT:



7. Program to implement Bresenham's circle drawing algorithm.

## SOURCE CODE:

```
#include<graphics.h>

#include<dos.h>

#include<stdio.h> void

drawcircle(int xc,int yc,int x,int y)

{

putpixel(xc+x,yc+y,RED);

putpixel(xc-x,yc+y,RED);

putpixel(xc+x,yc-y,RED);

putpixel(xc-x,yc-y,RED);

putpixel(xc+y,yc+x,RED);

putpixel(xc-y,yc+x,RED);

putpixel(xc+y,yc-x,RED);

putpixel(xc-y,yc-x,RED);

}

void circleBres(int xc,int yc,int r)

{

int x=0,y=r; int

d=3-2*r;
```

```
drawcircle(xc,yc,
x,y); while(y>=x)
{
x++;
if(d>0)
{
y--;
d=d+4*(x-y)+10;
}
else
d=d+4*x+6;
drawcircle(xc,yc,x,y);
//delay(5);
}
}
int main()
{
int xc,yc,r;
int gd=DETECT,gm;
initgraph(&gd,&gm,"C:\\TURBOC3\\B
GI"); printf("Enter x and y " );
scanf("%d%d",&xc,&yc);
```

```
printf("\nEnter radius: ");
scanf("%d",&r); if(r<=0)
{
printf("Radius cannot be zero or negative");
}
circleBres(xc,y
c,r); getch();
return 0;
}
```

## OUTPUT:

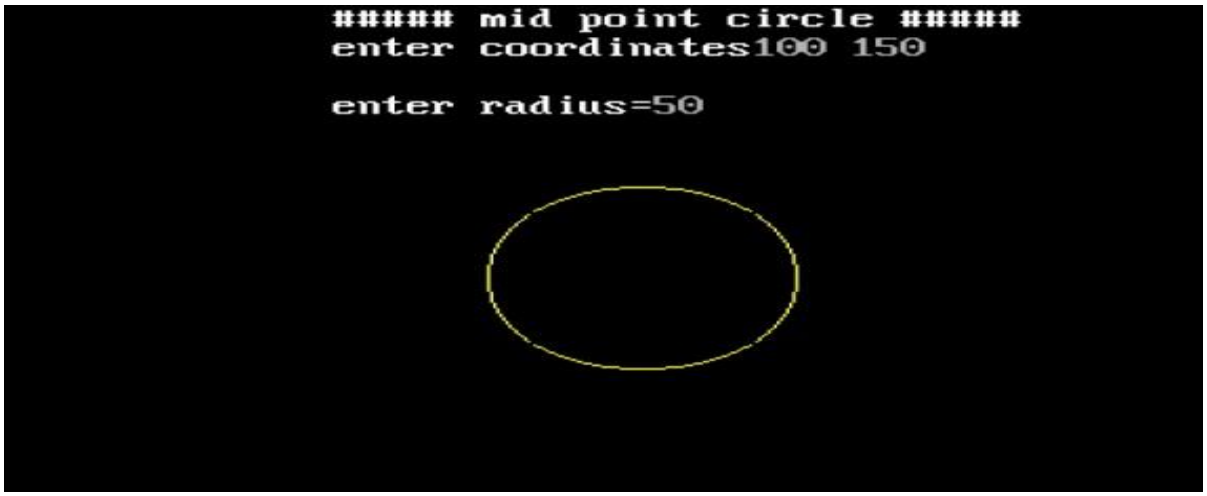
8. Program to implement mid point circle drawing algorithm.

Code:

```
#include<graphics.h>
#include<stdio.h>
#include<conio.h>
void main()
{
int x,y,x_mid,y_mid,radius,dp,mid;
int g_mode,g_driver=DETECT;
clrscr();
initgraph(&g_driver,&g_mode,"c:\\turboC3\\bgi");
printf("##### mid point circle #####");
printf("\nenter coordinates");
scanf("%d %d",&x_mid,&y_mid);
printf("\nenter radius=");
scanf("%d",&radius);
x=0;
y=radius;
dp=1-radius;
do
{
putpixel(x_mid+x,y_mid+y,YELLOW);
putpixel(x_mid+y,y_mid+x,YELLOW);
putpixel(x_mid-y,y_mid+x,YELLOW);
```

```
putpixel(x_mid-x,y_mid+y,YELLOW);
putpixel(x_mid-x,y_mid-y,YELLOW);
putpixel(x_mid-y,y_mid-x,YELLOW);
putpixel(x_mid+y,y_mid-x,YELLOW);
putpixel(x_mid+x,y_mid-y,YELLOW);
if(dp<0)
{
dp+=(2*x)+1;
}
else{
y=y-1;
dp+=(2*x)-(2*y)+1;
}
x=x+1;
}while(y>x);
getch();
closegraph();
}
```

## OUTPUT:



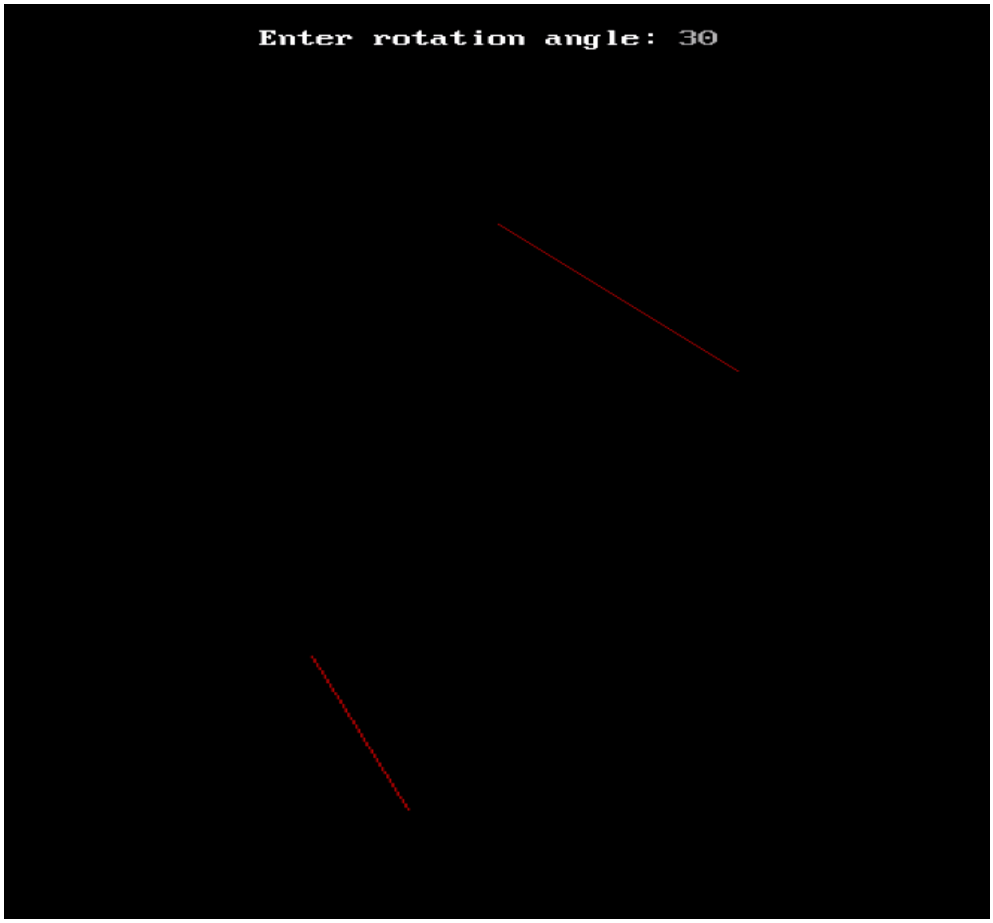
9. Program to show rotation of an object :

Source Code:

```
#include<stdio.h>
#include<graphics.h>
#include<math.h>
int main()
{
 int gd=0,gm,x1,y1,x2,y2;
 double s,c, angle;
 initgraph(&gd, &gm, "C:\\TURBOC3\\BGI");
 setcolor(RED);
 printf("Enter coordinates of line: ");
 scanf("%d%d%d%d",&x1,&y1,&x2,&y2);
 cleardevice();
line(x1,y1,x2,y2);
 getch();
printf("Enter rotation angle: ");
 scanf("%lf", &angle);
c = cos(angle *3.14/180);
 s = sin(angle *3.14/180);
 x1 = floor(x1 * c - y1 * s);
 y1 = floor(x1 * s + y1 * c);
 x2 = floor(x2 * c - y2 * s);
 y2 = floor(x2 * s + y2 * c);
cleardevice();
 line(x1, y1 ,x2, y2);
 getch();
closegraph();
return 0;
```

```
}
```

OUTPUT:

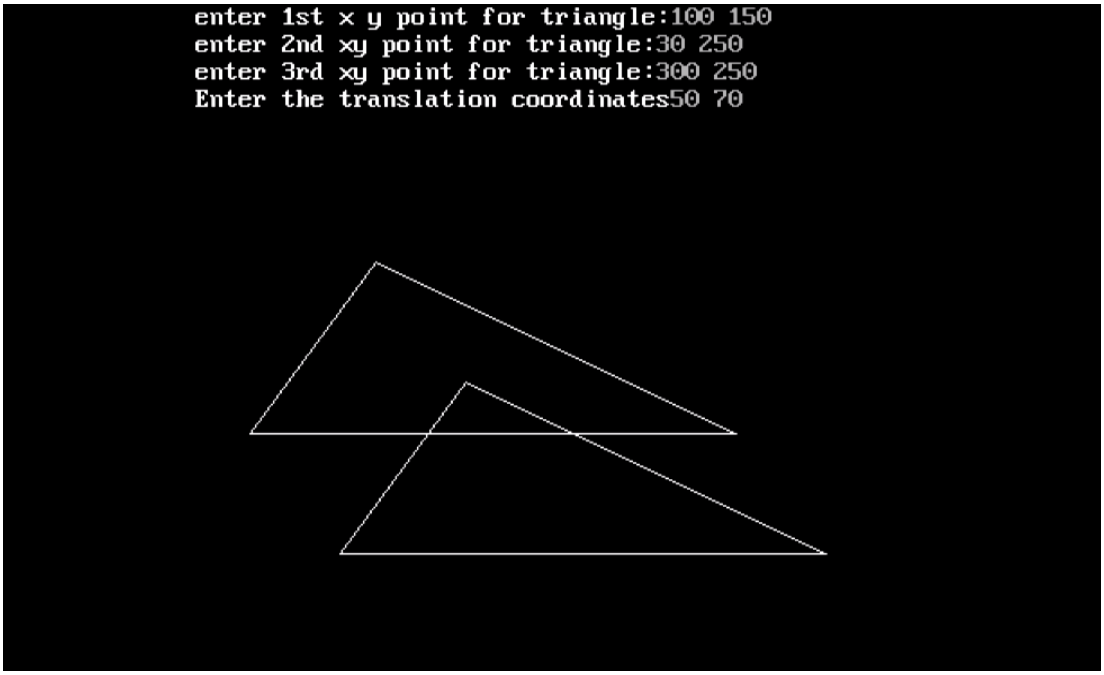




10. Program to show Translation:

## Source Code:

```
#include<stdio.h>
#include<graphics.h>
void main()
{
int gd=DETECT,gm;
int c,x1,x2,y1,y2,x3,y3,x,y,a1,a2,a3,b1,b2,b3;
clrscr();
initgraph(&gd,&gm,"C:\\TURBOC3\\BGI");
printf("enter 1st x y point for triangle:");
scanf("%d%d",&x1,&y1);
printf("enter 2nd xy point for triangle:");
scanf("%d%d",&x2,&y2);
printf("enter 3rd xy point for triangle:");
scanf("%d%d",&x3,&y3);
line(x1,y1,x2,y2);
line(x2,y2,x3,y3);
line(x3,y3,x1,y1);
printf("Enter the translation coordinates");
scanf("%d %d",&x,&y);
a1=x1+x;
b1=y1+y;
a2=x2+x;
b2=y2+y;
a3=x3+x;
b3=y3+y;
line(a1,b1,a2,b2);
line(a2,b2,a3,b3);
line(a3,b3,a1,b1);
getch();
closegraph();
}
```

OUTPUT:




11. Clipping:

Source code:

```
#include<graphics.h>
#include<conio.h>
#include<stdio.h>
#include<math.h>
void main()
{
int
rcode_begin[4]={0,0,0,0},rcode_end[4]={0,0,0,0},region_code[4
];
int W_xmax,W_ymax,W_xmin,W_ymin,flag=0;
```

```
float slope;
int x,y,x1,y1,i, xc,yc;
int gr=DETECT,gm;
initgraph(&gr,&gm,"C:\\TURBOC3\\BGI");
printf("\n****** Cohen Sutherlsnd Line Clipping algorithm
***********");
printf("\n Now, enter XMin, YMin =");

scanf("%d %d",&W_xmin,&W_ymin);
printf("\n First enter XMax, YMax =");
scanf("%d %d",&W_xmax,&W_ymax);
printf("\n Please enter intial point x and y= ");
scanf("%d %d",&x,&y);
printf("\n Now, enter final point x1 and y1= ");
scanf("%d %d",&x1,&y1);
cleardevice();
rectangle(W_xmin,W_ymin,W_xmax,W_ymax);
line(x,y,x1,y1);
line(0,0,600,0);
line(0,0,0,600);
if(y>W_ymax)  {
rcode_begin[0]=1;     // Top
flag=1 ;
}
if(y<W_ymin) {
rcode_begin[1]=1;          // Bottom
flag=1;
}
```

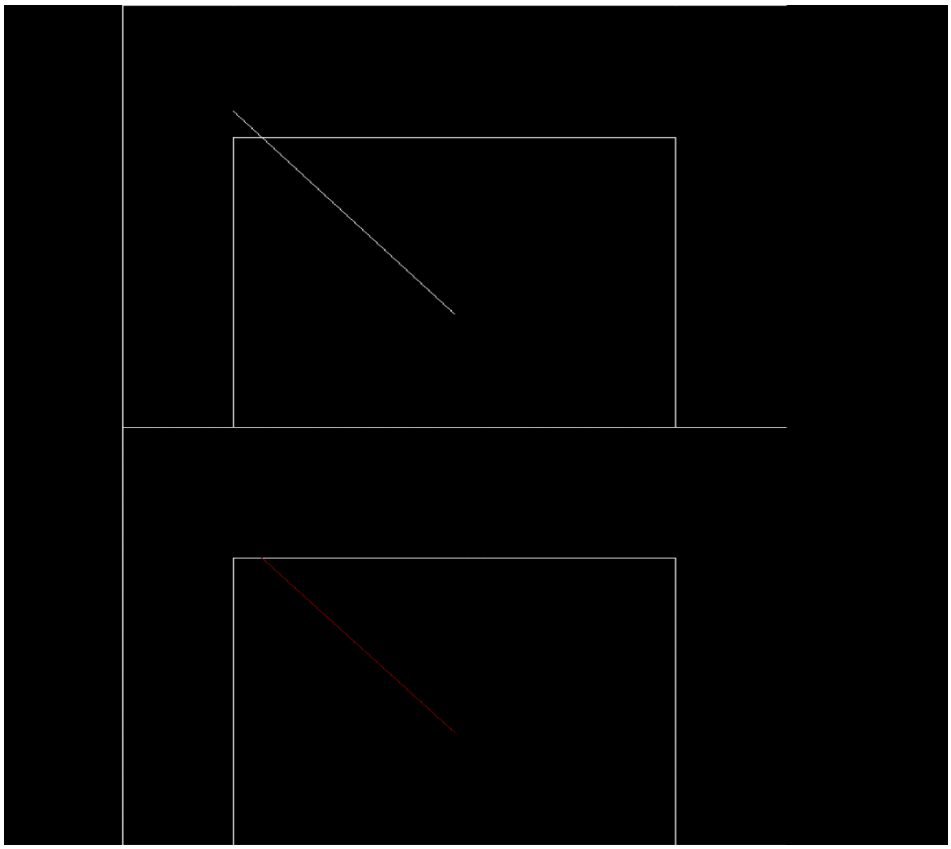
```
if(x>W_xmax)  {
rcode_begin[2]=1;          // Right
flag=1;
}
if(x<W_xmin)   {
rcode_begin[3]=1;          //Left
flag=1;
}

//end point of Line
if(y1>W_ymax){
rcode_end[0]=1;          // Top
flag=1;
}
if(y1<W_ymin) {
rcode_end[1]=1;          // Bottom
flag=1;
}
if(x1>W_xmax){
rcode_end[2]=1;          // Right
flag=1;
}
if(x1<W_xmin){
rcode_end[3]=1;          //Left
flag=1;
 }
if(flag==0)
{
```

```
printf("No need of clipping as it is already in window");
}
flag=1;
for(i=0;i<4;i++){
region_code[i]= rcode_begin[i] && rcode_end[i] ;
if(region_code[i]==1)
 flag=0;
}
if(flag==0)
{
printf("\n Line is completely outside the window");
}
else{
slope=(float)(y1-y)/(x1-x);
if(rcode_begin[2]==0 && rcode_begin[3]==1)   //left
{
y=y+(float) (W_xmin-x)*slope ;
x=W_xmin;

}
if(rcode_begin[2]==1 && rcode_begin[3]==0)      // right
{
y=y+(float) (W_xmax-x)*slope ;
x=W_xmax;

}
if(rcode_begin[0]==1 && rcode_begin[1]==0)     // top
{
```

```
x=x+(float) (W_ymax-y)/slope ;
y=W_ymax;

}
if(rcode_begin[0]==0 && rcode_begin[1]==1)     // bottom
{
x=x+(float) (W_ymin-y)/slope ;
y=W_ymin;

}
// end points
if(rcode_end[2]==0 && rcode_end[3]==1)   //left
{
y1=y1+(float) (W_xmin-x1)*slope ;
x1=W_xmin;

}
if(rcode_end[2]==1 && rcode_end[3]==0)      // right
{
y1=y1+(float) (W_xmax-x1)*slope ;
x1=W_xmax;

}
if(rcode_end[0]==1 && rcode_end[1]==0)     // top
{
x1=x1+(float) (W_ymax-y1)/slope ;
y1=W_ymax;
```

```
}
if(rcode_end[0]==0 && rcode_end[1]==1)     // bottom
{
x1=x1+(float) (W_ymin-y1)/slope ;
y1=W_ymin;

}
}
delay(1000);
clearviewport();
rectangle(W_xmin,W_ymin,W_xmax,W_ymax);
line(0,0,600,0);
line(0,0,0,600);
setcolor(RED);
line(x,y,x1,y1);
getch();
closegraph();
}
```

OUTPUT:



# JAVA PROGRAMS

1. Multithreading:

Source Code:

```
class multithreads extends Thread {
public void run() {
for (int i = 1; i < 5; i++) {
 try {
 Thread.sleep(500);
 } catch (InterruptedException e) {
 System.out.println(e);
  }
  System.out.println(i);
   }
   }
    public static void main(String args[]) {
    multithreads t1 = new multithreads();
    multithreads t2 = new multithreads();
```

```
        t1.run();
        t2.run();
    }
}
```

OUTPUT:

```
run:
1
2
3
4
1
2
3
4
BUILD SUCCESSFUL (total time: 9 seconds)
```

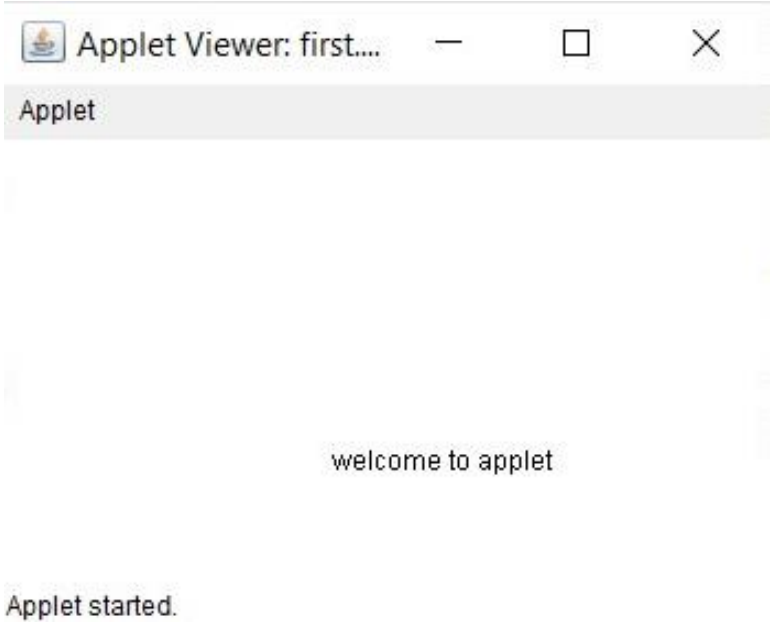
2. Program to implement Simple applet using appletViewer.

Source Code:

```
import java.applet.Applet;
import java.awt.Graphics;
public class first extends Applet{
@Override
```

```
public void paint(Graphics g){
g.drawString("welcome to applet",150,150);
}
}
/*
<applet code="First.class" width="300" height="300">
</applet>
*/
}
```

## OUTPUT:



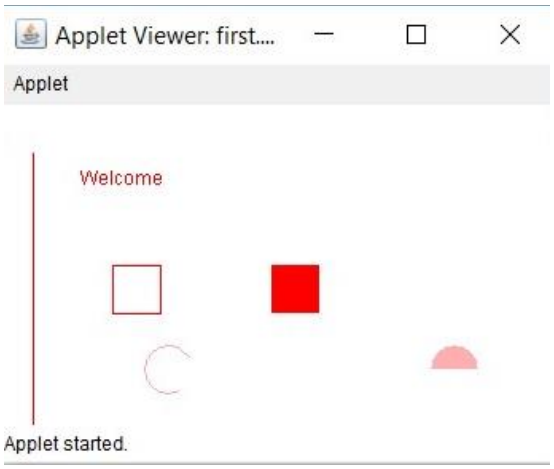

## 3. Program to implement Applets:

### Code:

```
import java.applet.Applet;
import java.awt.Color;
import java.awt.Graphics;
```

```
public class first extends Applet{
@Override
public void paint(Graphics g){
g.setColor(Color.red);
g.drawString("Welcome",50, 50);
g.drawLine(20,30,20,300);
g.drawRect(70,100,30,30);
g.fillRect(170,100,30,30);
g.drawOval(70,200,30,30);
g.setColor(Color.pink);
g.fillOval(170,200,30,30);
g.drawArc(90,150,30,30,30,270);
g.fillArc(270,150,30,30,0,180);
}
}
/*
<applet code="First.class" width="300" height="300">
</applet>
*/
}
```

## OUTPUT:

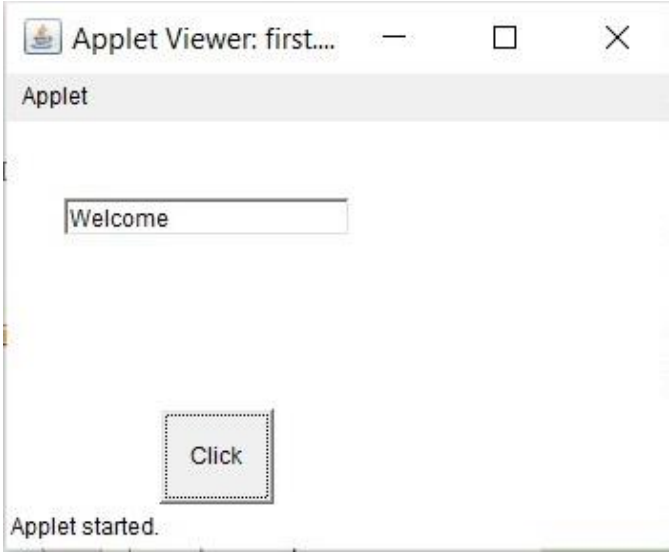


4. Program to implement Applets Event Handling.

Code:

```
import java.applet.Applet;
import java.awt.*;
import java.awt.event.ActionEvent;
import java.awt.event.ActionListener;
public class first extends Applet implements ActionListener{
Button b;
TextField tf;
public void init(){
tf=new TextField();
tf.setBounds(30,40,150,20);
b=new Button("Click");
b.setBounds(80,150,60,50);
add(b);add(tf);
b.addActionListener(this);
setLayout(null);
}
@Override
public void actionPerformed(ActionEvent e){
tf.setText("Welcome");
} }
```

OUTPUT:




5. <u>Program – Quick sort time analysis Source Code:</u>

```
#include <bits/stdc++.h>

#include <time.h>

using namespace std;

int partition(int arr[], int l, int h);

void quickSort(int arr[], int l, int h)

{

if (l < h)

{
```

```
int pivot = partition(arr, l, h) ;
quickSort(arr, l, pivot - 1);
quickSort(arr, pivot + 1, h);
}
}
int partition(int arr[], int startIndex, int endIndex)
{
int pointer=startIndex;
int pivotValue=arr[endIndex];
for(int i=startIndex+1;i<endIndex;i++){
if(arr[pointer]<=pivotValue)
pointer++;
if(arr[i]<=pivotValue){
int temp=arr[i];
arr[i]=arr[pointer];
arr[pointer]=temp;
}
}
if(arr[pointer]<=pivotValue)
pointer++;
int temp=arr[pointer];
arr[pointer]=arr[endIndex];
```

```
arr[endIndex]=temp;
return pointer;
}
void printArray(int arr[], int n)
{
for (int i = 0; i < n; i++)
{
cout << arr[i] << " ";
}
}
int main()
{
cout << "Enter the size : -\n" ;
int arr[1000] ;
for (int i=0;i<1000;i++){
arr[i]=i;
}
clock_t start, end ;
start = clock();
quickSort(arr, 0, 10);
end = clock() ;
double tim_takn = double(end-start)/double(CLOCKS_PER_SEC/1000 );
```

```
cout << "array after sorting :\n" ;

printArray(arr, 11);

cout<<"Time taken"<<fixed<<tim_takn;

return 0;

}
```

## OUTPUT:

```
Enter the size : -
array after sorting :
0 1 2 3 4 5 6 7 8 9 10 Time taken0.000000
--------------------------------
Process exited after 0.1729 seconds with return value 0
Press any key to continue . . .
```