آموزش ساخت بازی

دو بعدی در جاوا

اثری از

رحمان زارعی

به نام خدا

امام صادق علیه‌السلام :

إنَّ لِكُلِّ شَيءٍ زَكاةً ، وزَكاةُ العِلمِ أن يُعَلِّمَهُ أهلَهُ .

هرچیزی زکاتی دارد و زکات دانش، آموختن آن است به اهلش.

تقدیم به هموطنان عزیزم به خصوص مناطق محروم

زبان برنامه نویسی جاوا

آموزش ساخت بازی دو بعدی

جلسه اول

نویسنده : رحمان زارعی

جاوا را ساده،آسان و شیرین بنوشید!!!!

چکیده

سلام. روزتون بخیر. امیدوارم پرانرژی و عالی باشید. این اولین جلسه ای است که قصد داریم گامی محکم در جهت یادگیری روش ساخت بازی دو بعدی با زبان برنامه نویسی جاوا برداریم. ایجاد یک بازی با استفاده از زبان جاوا ابتدا ممکن است دشوار به نظر برسد،اما زمانی که شما از اصول اولیه پیروی کنید متوجه خواهید شد چقدر بازی نوشتن با جاوا ساده است! امروزه بسیاری از بازی ها برای پلتفرم های Android و iOS با استفاده از جاوا توسعه می یابد.

این را بدانید که اصول ساخت یک بازی در هر زبان برنامه نویسی ثابت می باشد مثلا در هر بازی با رفتاری به نام حرکت کاراکترهای بازی،کنترل کاراکترهای بازی مثلا با موس یا کیبورد و... سروکار داریم.

ما در این کتاب آموزشی قصد داریم شما را با مراحل ساخت یک بازی دو بعدی تحت ویندوز بصورت پروژه محور آشنا کنیم. بهترین روش برای یادگیری کدزدن ، کدبازی است!!! یعنی این که بجای توضیح اضافی برای هر مفهوم مثال کدنویسی آورده می شود!

قبل از شروع این آموزش بهتر از با مفاهیم اولیه جاوا، شی گرایی ، کار با فایل، چندنخی و گرافیک در جاوا آشنایی داشته باشید.

همه موارد بالا را ما در کانال و سایت آموزش داده ایم. برای درک بهتر آموزش ساخت یک بازی ساده را شروع می کنیم.

گام اول: تعریف مسئله و استخراج مفاهیم درون مسئله

هر پروژه برنامه نویسی یک **مسئله** دارد که برای **حل** آن به شما داده می شود و شما باید **مفاهیم** درون مسئله رو استخراج کنید و در نهایت **شروع به کدنویسی** کنید. گاهی هم یه عکس بهتون میدن میگن من برنامه ای شبیه به این چیزی که در تصویر هست میخوام! اون وقت شما باید خودتون با توجه به برداشت هایی که از تصویر میکنید برای خودتون مسئله بنویسید و در پایان مفاهیم مسئله نوشته شده رو حل کنید.

تعریف مسئله: بازی گرفتن تخم مرغ

در این بازی تعدادی تخم مرغ از بالا به پایین رها می شوند و کاربر با داشتن یک سبد باید تمامی تخم مرغ ها رو جمع آوری کند و اجازه ندهد حتی یک تخم مرغ روی زمین بیفتد.با جمع آوری هر تخم مرغ ۱۰ امتیاز به کاربر داده می شود، در صورت افتادن یک تخم مرغ بر روی زمین کاربر بازی را می بازد و Game Over می شود.

در این بازی تخم مرغ ها در موقعیت افقی تصادفی از بالا به پایین رها می شوند. کاربر با کلید های راست (➡) و چپ(⬅) کیبورد سبد را در موقعیت افقی جابه جا می کند به گونه ای که بتواند تخم مرغ های رها شده را بگیرد.

این بازی باید گرافیکی و زیبا باشد.

استخراج مفاهیم درون مسئله بازی گرفتن تخم مرغ

خب ما اینجا یک مسئله اماده داریم و قصد داریم مفاهیمی که باید آن را به شکل کدنویسی دربیاریم استخراج کنیم. کلیت مسئله این است که ما با یک بازی سر و کار داریم و اونم بازی گرفتن تخم مرغ هستش.

اولین گام حل مسئله بازی گرفتن تخم مرغ

اولین گام باید مفاهیم جاوایی که مورد نیاز برای ساخت بازی نظیر کلاس ها، متغیرها، متدها، اشیا، کلیدهای کنترل رویداد های بازی ، کار با فایل و.... را استخراج کنیم.

دومین گام حل مسئله بازی گرفتن تخم مرغ

دومین گام سراغ اصول اولیه ساخت بازی که بین همه پروژه های بازی سازی به زبان جاوا مشترک هستند می رویم. (جلوتر سراغ این اصول خواهیم رفت)

بررسی اولین گام حل مسئله بازی گرفتن تخم مرغ

متغیرهای درون مسئله رو با رنگ ابی هایلایت می کنیم.

کلاس های درون مسئله رو با رنگ سبز هایلایت می کنیم.

متدهای درون مسئله رو با رنگ زرد هایلایت می کنیم.

مفاهیم هایلایت شده مسئله:

در این بازی تعدادی تخم مرغ از بالا به پایین رها می شوند و کاربر با داشتن یک سبد باید تمامی تخم مرغ ها رو جمع آوری کند و اجازه ندهد حتی یک تخم مرغ روی زمین بیفتد.با جمع آوری هر تخم مرغ ۱۰ امتیاز به کاربر داده می شود، در صورت افتادن یک تخم مرغ بر روی زمین کاربر بازی را می بازد و Game Over می شود.

در این بازی تخم مرغ ها در موقعیت افقی تصادفی از بالا به پایین رها می شوند. کاربر با کلید های راست (➡) و چپ(⬅) کیبورد سبد را در موقعیت افقی جابه جا می کند به گونه ای که بتواند تخم مرغ های رها شده را بگیرد.

این بازی باید گرافیکی و زیبا باشد.

خب ما در این مسئله چهار کلاس به نام های بازی ، تخم مرغ ، کاربر و سبد داریم که هر کدام از متغیرها و متدهای هایلایت شده به کلاس های مذکور مربوط می شود. ما باید هر کلاس با متغیرها و متدهایی که دارد را با کدنویسی پیاده سازی کنیم.

قبل از پیاده سازی کلاس های این مسئله ، اصول ثابتی که در ساخت هر بازی نیاز است را در ادامه بررسی می کنیم بعد به پیاده سازی پروژه بازی گرفتن تخم مرغ می پردازیم.

بررسی دومین گام حل مسئله بازی گرفتن تخم مرغ

ما در اینجا اصولی که برای ساخت یک بازی گرافیکی در جاوا به ما کمک می کنند را بررسی می کنیم.

اصل اول:یک پنجره ساده

در اصل اول باید محیطی که قراره بازی در آن نمایش داده شود یا Board بازی را بسازیم.برای این کار یک کلاس با نام Board ایجاد می کنیم:

```
package javalikeGameTutorial;

public class Board {

}
```

- در کد بالا ما یک کلاس با نام Board درون پکیج javalikeGameTutorial ایجاد کرده ایم.

کلاس JPanel:

- JPanel مانند بوم نقاشی است که می توانیم اجزای گرافیکی و درنهایت کل بازی را روی آن ترسیم کنیم.پس لازم است کلاس Board که وظیفه نمایش محیط بازی را برعهده دارد کلاس JPanel را به ارث ببرد تا به ویژگی ها و رفتارهای این کلاس دسرسی پیدا کند.

```
package javalikeGameTutorial;

import javax.swing.JPanel;

public class Board extends JPanel {

}
```

- برای استفاده از کلاس JPanal در برنامه خود باید پکیج زیر را در کلاس خود import کنید:

```
import javax.swing.JPanel;
```

- در کل ما برای ایجاد یک برنامه گرافیکی با استفاده از جاوا لازم است دو پکیج زیر رو در کلاس های مربوطه import کنیم:

```
import javax.swing.*;
import java.awt.*;
```

- خب حالا برای این که بازی قابلیت اجرا داشته باشد نیاز به کلاسی که دارای متد main می باشد هستیم. برای این کار یک کلاس با نام Main که دارای متد main می باشد را درون پکیج javalikeGameTutorial بصورت زیر ایجاد می کنیم:

```
package javalikeGameTutorial;

public class Main {

   public static void main(String[] args) {

   }

}
```

کلاس Main برای اجرا و نمایش کلاس Board ما که کلاس JPanel را به ارث برده است نیاز به یک JFrame دارد.پس برای این کار باید کلاس Main کلاس JFrame را به ارث ببرد:

```
package javalikeGameTutorial;

import javax.swing.JFrame;

public class Main extends JFrame {

      public static void main(String[] args) {

      }

}
```

- خب حالا سازنده کلاس Main را تعریف می کنیم :

```
package javalikeGameTutorial;

import javax.swing.JFrame;
```

```
public class Main extends JFrame {

	public Main() {
	}

	public static void main(String[] args) {

	}

}
```

- حالا برای افزودن کلاس Board خود به JFrame متد add کلاس JFrame را صدا می زنیم و شی از نوع کلاس Board را جایگزین پارامتر این متد می کنیم:

```
package javalikeGameTutorial;

import javax.swing.JFrame;

public class Main extends JFrame {

	public Main() {

		add(new Board());
	}

	public static void main(String[] args) {

	}

}
```

خب حالا عنوانی را برای پنجره فریم خود تنظیم می کنیم:

```
package javalikeGameTutorial;

import javax.swing.JFrame;

public class Main extends JFrame {

	public Main() {

		add(new Board());
```

```
            setTitle("Game");
      }

      public static void main(String[] args) {

      }

}
```

برای این که بعد از کلیک کردن روی دکمه close یا همان ضربدر قرمز رنگ گوشه سمت راست پنجره فریم برنامه به طور کامل بسته شود از دستور ثابت زیر استفاده می کنیم:

```
setDefaultCloseOperation(JFrame.EXIT_ON_CLOSE);
```

حالا در کلاس Main خواهیم داشت:

```
package javalikeGameTutorial;

import javax.swing.JFrame;

public class Main extends JFrame {

      public Main() {

            add(new Board());
            setTitle("Game");
            setDefaultCloseOperation(JFrame.EXIT_ON_CLOSE);
      }

      public static void main(String[] args) {

      }

}
```

- برای قرار دادن پنجره فریم در مرکز صفحه نمایش مانیتور از دستور زیر استفاده می کنیم:

```
setLocationRelativeTo(null);
```

- با متد زیر اندازه و ابعاد پنجره فریم خود را تنظیم می کنیم:

```
setSize(400,400);
```

- برای این که کاربر نتواند اندازه فریم رو دستکاری و کم و زیاد کند از متد زیر استفاده کرده و مقدار پارامتر آن را false قرار می دهیم:

```
setResizable(false);
```

- برای نمایش فریم و تمام اجزای گرافیکی برنامه از متد زیر استفاده می کنیم و مقدار پارامتر آن را true قرار می دهیم:

```
setVisible(true);
```

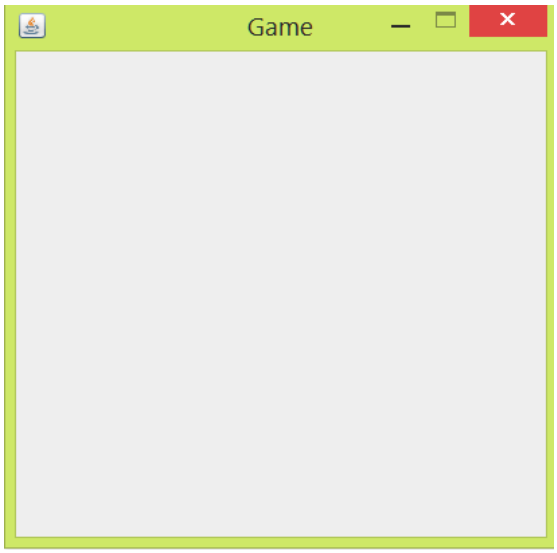
حال تا اینجا کلاس Main به صورت زیر خواهد بود:

```
package javalikeGameTutorial;

import javax.swing.JFrame;

public class Main extends JFrame {

public Main() {

add(new Board());
setTitle("Game");
setDefaultCloseOperation(JFrame.EXIT_ON_CLOSE);
setLocationRelativeTo(null);
setSize(400,400);

setResizable(false);

setVisible(true);

}

public static void main(String[] args) {

}

}
```

حالا در متد main کلاس Main یک شی از کلاس Main ایجاد می کنیم:

```
package javalikeGameTutorial;

import javax.swing.JFrame;

public class Main extends JFrame {

	public Main() {

		add(new Board());
		setTitle("Game");
		setDefaultCloseOperation(JFrame.EXIT_ON_CLOSE);
		setLocationRelativeTo(null);
		setSize(400,400);
		setResizable(false);
		setVisible(true);
	}

	public static void main(String[] args) {
		Main m = new Main();
	}

}
```

- حال برنامه بالا رو اجرا می کنیم: خروجی تصویر(۱)
- توجه داشته باشید برنامه بالا تا اینجا شامل یک کلاس با نام Board و یک کلاس با نام Main می باشد.



تصویر(۱)

اصل دوم:نقاشی روی بوم

تا الان شما با محیطی که قرار است بازی در آن اجرا شود آشنا شدید. سعی کنید به کدها و شبه کدهایی که تا الان دیدید و بعداً خواهید دید بسیار دقت کنید و سعی کنید اونارو بفهمید چون این کدها پایه های نوشتن پروژه های گرافیکی هستند و با یادگیری آنها پروژه هایتان را راحت تر می توانید انجام دهید.

رسم اشکال هندسی نظیر مستطیل ، دایره و بیضی تو پر و خالی و خط :

در مرحله قبل ما طرز ساخت یک پنجره را به کمک جاوا یادگرفتیم . جاوا دارای توابع گسترده ای برای رسم اشکال مختلف در اختیار دارد که به JAVA 2D API معروف هستند . این توابع و کلاس ها شامل ابزار نقاشی ، کار با تصاویر و رنگ و ... هستند . در این قسمت ما می خواهیم با این توابع و ابزار جاوا بیشتر آشنا شویم و باهم اشکال ذکر شده رو به کمک API های جاوا ایجاد کنیم. (البته ما تمام این اشکال گرافیکی ، متدها و... را در مبحث کار با گرافیک در کانال و سایت خود بصورت مجزا آموزش داده ایم.)

نکته مهم: برای درک بهتر جلسه ۲۵ آموزش گرافیک در جاوا مبحث کلاس Graphics را مطالعه کنید.

به کد برنامه زیر توجه کنید: برنامه زیر از دو کلاس به نام `Board` و `Main` تشکیل شده است.برای اجرای برنامه باید کلاس `Main` را `run` (اجرا) کنیم:

کلاس `Board` : نام و فرمت فایل `Board.java`

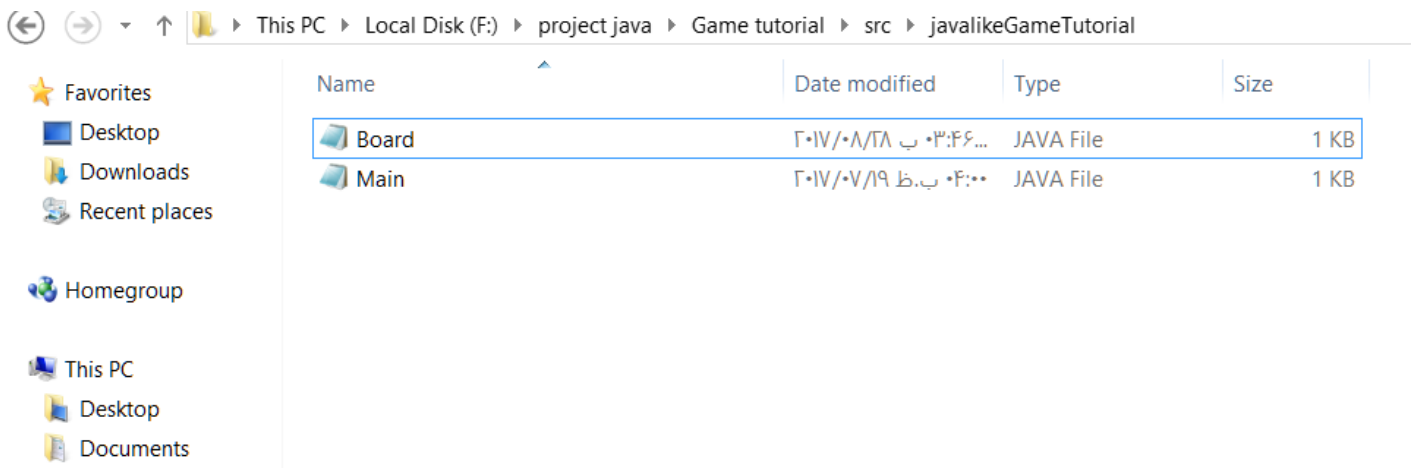
```
package javalikeGameTutorial;

import javax.swing.JPanel;
import javax.swing.*;
import java.awt.*;

public class Board extends JPanel {
      int Width = 200, Height = 50, x = 10, y = 10;
      int x2 = 50, y2 = 100, Width2 = 200, Height2 = 50;
      int x3 = 50, y3 = 200, Dim = 70;
      int x4 = 50, y4 = 300, Width4 = 200, Height4 = 50;

      public Board() {

      }
```

```
	public void paint(Graphics g) {
		super.paint(g);
		Graphics2D g2 = (Graphics2D) g;

		g2.setRenderingHint(RenderingHints.KEY_ANTIALIASING,
				RenderingHints.VALUE_ANTIALIAS_ON);
		g2.setColor(Color.blue);
		g2.drawRect(x, y, Width, Height);
		g2.setColor(Color.green);
		g2.fillRect(x2, y2, Width2, Height2);
		g2.setColor(Color.magenta);
		g2.fillOval(x3, y3, Dim, Dim);
		g2.setColor(Color.red);
		g2.fillOval(x4, y4, Width4, Height4);

	}

}
```

کلاس Main : نام و فرمت فایل Main.java

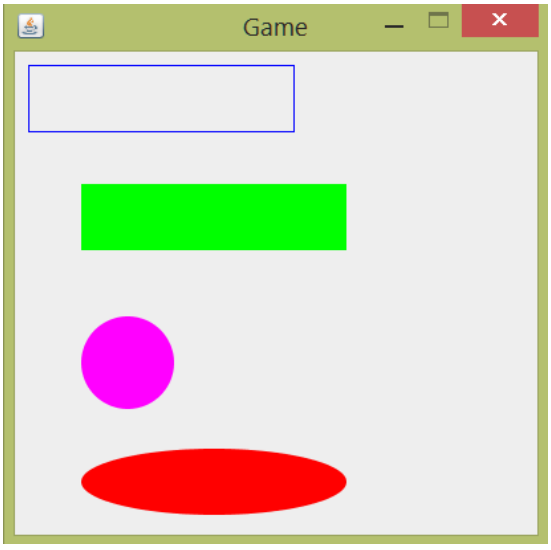
```
package javalikeGameTutorial;

import javax.swing.JFrame;

public class Main extends JFrame {

	public Main() {

		add(new Board());
		setTitle("Game");
		setSize(400,400);
		setDefaultCloseOperation(JFrame.EXIT_ON_CLOSE);
		setLocationRelativeTo(null);
		setResizable(false);
		setVisible(true);
	}

	public static void main(String[] args) {
		Main m = new Main();
	}

}
```

- منظور از نام و فرمت فایل همون فایل های جاوایی است که حاوی کدهای کلاس برنامه میباشد. تصویر(۲)



تصویر(۲)

- در تصویر(۲) فایل دو کلاس `Board` و `Main` در پکیج `javalikeGameTutorial` موجود در فولدر `src` پروژه `Game tutorial` را مشاهده می کنید.
- برای اجرای برنامه فوق کافیست کلاسی که دارای متد `main` می باشد را`run` کنیم. در این برنامه متد `main` درون کلاس `Main` قرار دارد.پس کلاس `Main` را اجرا می کنیم. خروجی بصورت تصویر(۳) خواهد بود:



تصویر(۳)

توضیحات برنامه فوق :

همان طور که گفتیم این برنامه تا اینجا از دو کلاس به نام Board و Main تشکیل شده است. که هر کدام را به صورت جداگانه بررسی می کنیم:

بررسی کلاس Board :

```
package javalikeGameTutorial;
```

- کلاس ما درون این پکیج قرار گرفته است.

```
import javax.swing.JPanel;
import javax.swing.*;
import java.awt.*;
```

- کتابخانه ها یا پکیج هایی که برای نوشتن این برنامه گرافیکی نیاز داریم را import می کنیم.

```
public class Board extends JPanel {
```

- چون قصد داریم اشکال هندسی و گرافیکی در برنامه خود رسم کنیم نیاز داریم کلاس خود را به کلاس JPanel کنیم. با این کار کلاس ما فرزند کلاس JPanel می شود. extends

```
int Width = 200, Height = 50, x = 10, y = 10;
      int x2 = 50, y2 = 100, Width2 = 200, Height2 = 50;
      int x3 = 50, y3 = 200, Dim = 70;
      int x4 = 50, y4 = 300, Width4 = 200, Height4 = 50;
```

- ما در اینجا سری متغیر تعریف کرده ایم. متغیرهای x و y مختصات اشکال هندسی ، متغیر های Width عرض و Height ارتفاع (طول) اشکال هندسی را مشخص می کند. و متغیر Dim قطر برای بیضی و دایره استفاده می کنیم.
- پس توجه داشته باشید هر وقت خواستید یک برنامه بنویسید که اشکال هندسی را نمایش دهد باید برای این اشکال مختصات ، طول و عرض ، قطر و... در نظر بگیرید.
- اولین اصل یک بازی همون نقاشی کاراکترهای بازی هست که هر کاراکتر دارای **مختصاتی**(x,y) است که موقعیت ان را در صفحه بازی مشخص می کند و همچنین هر کاراکتر بازی دارای **ابعاد(طول و عرض)** نیز می باشد.

```
public void paint(Graphics g) {
            super.paint(g);
            Graphics2D g2 = (Graphics2D) g;

            g2.setRenderingHint(RenderingHints.KEY_ANTIALIASING,
                        RenderingHints.VALUE_ANTIALIAS_ON);
            g2.setColor(Color.blue);
            g2.drawRect(x, y, Width, Height);
            g2.setColor(Color.green);
            g2.fillRect(x2, y2, Width2, Height2);
            g2.setColor(Color.magenta);
            g2.fillOval(x3, y3, Dim, Dim);
            g2.setColor(Color.red);
            g2.fillOval(x4, y4, Width4, Height4);

      }
```

نکته: دوباره تاکید میکنم حتما جلسه ۲۵ آموزش کار با گرافیک موجود در کانال javalike را مطالعه کنید.

```
public void paint(Graphics g) {
```

- به زبان ساده برای نقاشی و رسم اشکال اشکال گرافیکی، متن ، تصویر و... از این متد استفاده می کنیم.
- پارامتر g یک شیء از کلاس Graphics است که یک کلاس انتزاعی است.برای رسم اشکال گرافیکی به این کلاس نیاز داریم.

```
super.paint(g);
```

- وقتی که شما دستور بالا را پیاده سازی می کنید، متد **paint کلاس پدر** کلاس ما صدا زده می شود! (توضیحات بیشتر جلسه ۲۵ آموزش کار با گرافیک در جاوا موجود در کانال javalike)

```
Graphics2D g2 = (Graphics2D) g;
```

- کلاس Graphics2D یک کنترل را در سطح بالا برای رسم شکل ایجاد می‌کند که ما برای آنکه بتوانیم از متدها و ویژگی های کلاس Graphics2D استفاده کنیم بایست یک تبدیل نوع (cast) بر روی شیء g انجام دهیم.

```
g2.setRenderingHint(RenderingHints.KEY_ANTIALIASING,
                        RenderingHints.VALUE_ANTIALIAS_ON);
```

- برای روان سازی خطوط رسم استفاده می شود.

```
1. g2.setColor(Color.blue);
2. g2.drawRect(x, y, Width, Height);
3. g2.setColor(Color.green);
4. g2.fillRect(x2, y2, Width2, Height2);
5. g2.setColor(Color.magenta);
6. g2.fillOval(x3, y3, Dim, Dim);
7. g2.setColor(Color.red);
8. g2.fillOval(x4, y4, Width4, Height4);
```

۱. برای رنگ کردن اشکال هندسی متد `setColor` را با شی `g2` که از نوع `Graphics2D` می باشد را صدا زده ایم. و یکی از پارامتر های مربوط به رنگ مورد نظر را درون متد `setColor` استفاده می کنیم.در اینجا از رنگ آبی استفاده شده است.

۲. برای رسم مستطیل تو خالی: (اگر عرض و ارتفاع را یکسان بدهیم مربع ترسیم میشود). `x,y` مختصات قرار گیری در صفحه و `Width, Height` ابعاد شکل هندسی که در اینجا مستطیل توخالی هست را تشکیل می دهد.

۳. برای شکل بعد رنگ سبز را تنظیم کرده ایم.

۴. برای رسم مستطیل تو پر از این دستور استفاده می شود.

۵. برای شکل بعد رنگ بنفش را انتخاب کرده ایم.

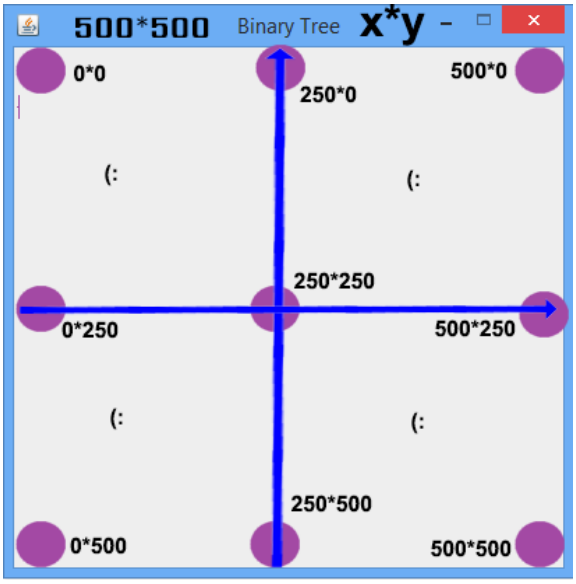
۶. برای رسم بیضی استفاده می شود: (اگر قطرهای بیضی را یک اندازه بدهیم دایره ترسیم می شود)

۷. برای شکل بعدی رنگ قرمز را انتخاب کرده ایم.

بررسی کلاس `Main` :

- دستورات درون کلاس `Main` را در همین جلسه آموزشی بررسی کرده ایم و چون دستور جدیدی به ان اضاف نکردیم از آن میگذریم.

نکته: قبل از رسم اشکال به این نکته دقت کنید که هر فریم طول و عرضی دارد و مختصاتی که اشکال درون آن قرار می‌گیرند به اینگونه هستند: `x`های فریم ازسمت چپ از صفر شروع می¬شوند تا به عرض فریم برسند و `y` فریم از بالا از صفر شروع شده تا این که به پایین(طول) فریم می‌رسد. در کل فریم ما مانند دستگاه مختصات یک `x` ای دارد و یک `y` ، با این تفاوت که `x` و `y` ها به ترتیب از چپ و بالا صفر هستند. تصویر زیر گویاست:تصویر(۴)



تصویر(۴) مختصات پیکسل ها در یک فریم

چند متد مهم

برای به دست آوردن طول و عرض پنجره(فریم) از دستور زیر در کلاسی که فریم به کار برده شده استفاده می کنیم:

```
getWidth()
getHeight()
```

برای تنظیم رنگ اشکال از دستور زیر استفاده می‌شود:

```
g2.setColor;(.........)
```

که بجای نقطه چین اسم رنگ رو می‌نویسیم. برای مثال:

```
g2.setColor(Color.red);
```

امیدوارم تا اینجا با مقدمات کار با گرافیک جاوا آشنا شده باشید و از کار و کد زدن با آن لذت برده باشید! چطور بود؟! فک کنم گرافیک جاوا دید شمارو در مورد برنامه نویسی عوض کرده ;-) . شاید بگید این رسما و اشکال به چه دردمون میخوره! آیا گرافیک جاوا همین بود؟چطور با اینا بازی بنویسیم؟! صبر داشته باشید :-)

برای پاسخ این سوال و سوالای دیگه با ما در گام های بعدی کار با گرافیک جاوا همراه شید!

گام سوم: تصاویر متحرک می‌شوند

در دو گام پیشین با مقدمات برنامه نویسی جاوا و همچنین مفاهیمی مانند گرافیک در جاوا آشنا شدید . در این گام با مفهوم انیمیشن و حرکت که یکی دیگر از مفاهیم پرکاربرد و مهم در ساخت برنامه‌های گرافیکی است آشنا خواهید شد. در ابتدا باید ببینیم که اصلا انیمیشن چیست؟

در واقع حرکت سریع تصاویر بصورت متوالی و پشت سر هم را انیمیشن گویند. ما قصد داریم با حرکت دایره بر روی Board (صفحه بازی که در جلسه قبل ایجاد کردیم) یک انیمیشن ایجاد کنیم و برای ایجاد انیمیشن از **مبحث چندنخی** در جاوا استفاده می کنیم.

نکته: ما در این آموزش همین جور داریم برنامه ساده قبل رو گسترش می دهیم.

کد زیر انیمیشن حرکت یک دایره هست که از گوشه بالا سمت چپ به صورت مایل به سمت راست - پایین حرکت می کند.
قبل از کد این نکته را یادآور شویم که ما همان کلاس های قبلی، یعنی Board را توسعه و تغییر داده ایم و تا اینجا برای اجرا از کلاس Main بدون تغییر استفاده می‌کنیم.

کلاس Board:

```
package javalikeGameTutorial;

import javax.swing.*;
import java.awt.*;

public class Board extends JPanel implements Runnable {

      int x3 = 0, y3 = 0, Dim = 70;
      int xa = 1, ya = 1;
      private Thread animator;

      public void move() {

            x3 = x3 + xa;
            y3 = y3 + ya;

      }

      public void paint(Graphics g) {
```

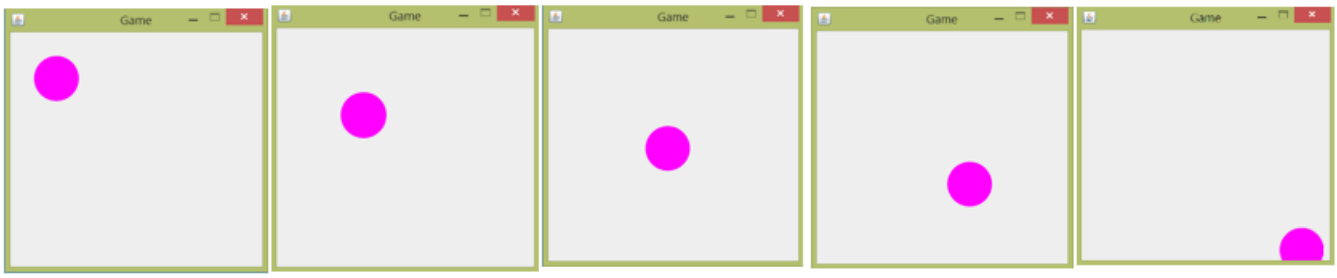
```
            super.paint(g);
            Graphics2D g2 = (Graphics2D) g;

            g2.setRenderingHint(RenderingHints.KEY_ANTIALIASING,
                        RenderingHints.VALUE_ANTIALIAS_ON);
            g2.setColor(Color.magenta);
            g2.fillOval(x3, y3, Dim, Dim);

      }

      public void addNotify() {
            super.addNotify();
            animator = new Thread(this);
            animator.start();
      }

      public void run() {

            while (true) {

                  repaint();
                  move();
                  try {
                        Thread.sleep(10);
                  } catch (InterruptedException e) {

                        e.printStackTrace();
                  }

            }
      }
}
```

کلاس Main :

```
package javalikeGameTutorial;

import javax.swing.JFrame;

public class Main extends JFrame {

      public Main() {

            add(new Board());
```

```
            setTitle("Game");
            setSize(400,400);
            setDefaultCloseOperation(JFrame.EXIT_ON_CLOSE);
            setLocationRelativeTo(null);
            setResizable(false);
            setVisible(true);
      }

      public static void main(String[] args) {
            Main m = new Main();
      }

}
```

خروجی: یک دایره صورتی رنگ متحرک می باشد که از گوشه بالا سمت چپ به صورت مورب حرکت می کند و از گوشه پایین سمت راست از صفحه خارج می شود: تصویر(۵)



تصویر(۵)

```
package javalikeGameTutorial;
```

- کلاس های ما درون پکیج `javalikeGameTutorial` قرار گرفته اند.

```
import javax.swing.*;
import java.awt.*;
```

- برای ساخت یک برنامه گرافیکی با جاوا `import` کردن دو پکیج کتابخانه ای جاوا الزامی است.

```
public class Board extends JPanel implements Runnable {
```

- همان طور که از قبل گفتیم برای ساخت انیمیشن در جاوا نیاز به استفاده از مبحث چندنخی در جاوا داریم. چون قراره حرکت شکل هندسی ما یعنی دایره روی کلاس Board نمایش داده شود کلاس Board را implements به اینترفیس Runnable می کنیم.

```
int x3 = 0, y3 = 0, Dim = 70;
```

- x3 و y3 مختصات قرار گیری شکل دایره ما در صفحه Board را مشخص می کند.
- Dim مشخص کننده قطر دایره ما می باشد.

```
int xa = 1, ya = 1;
```

این دو متغیر رو به خاطر بسپارید، جلوتر کاربردشو بهتون می گم.

```
private Thread animator;
```

- یک شی از نوع کلاس Thread ایجاد می کنیم.

```
    public void move() {

        x3 = x3 + xa;
        y3 = y3 + ya;

    }
```

- چون قراره شکل دایره ای ما در صفحه حرکت کند نیاز به پیاده سازی یک متد به نام move داریم.این متد نوع حرکت شکل ما در صفحه را مشخص می کند. اینجا دلخواه است دیگه، مثلا با توجه به نیاز می توانیم کاری کنیم که شکل ما حرکت در راستای محور افقی یا عمودی یا مورب یا سایر حرکت ها مثل حرکت دایره ای شکل یا حرکت پرتابه ای و.. داشته باشد.
- در اینجا با توجه به دستورات درون متد move ، دایره ما به صورت مورب حرکت می کند.
- xa و ya مقادیر ثابتی هستند که با مختصات شکل دایره ای ما جمع می شوند و مختصات شکل ما را تغییر می دهند و باعث انتقال شکل ما از یک مختصات خاص به یک مختصات دیگر در صفحه می شوند.

```
public void paint(Graphics g) {
        super.paint(g);
        Graphics2D g2 = (Graphics2D) g;

        g2.setRenderingHint(RenderingHints.KEY_ANTIALIASING,
                RenderingHints.VALUE_ANTIALIAS_ON);
        g2.setColor(Color.magenta);
        g2.fillOval(x3, y3, Dim, Dim);

    }
```

- در اینجا شکل دایره ای خود را با استفاده از متد paint رسم می کنیم.

```
super.paint(g);
        Graphics2D g2 = (Graphics2D) g;

        g2.setRenderingHint(RenderingHints.KEY_ANTIALIASING,
                RenderingHints.VALUE_ANTIALIAS_ON);
```

- دوستان به خودتون سخت نگیرید دستورات بالا در هر برنامه گرافیکی ثابت هستند و باید ازشون استفاده کنید.

```
g2.setColor(Color.magenta);
          g2.fillOval(x3, y3, Dim, Dim);
```

- رنگ شکل ما صورتی هستش.
- یک دایره توپر با مختصات x3 و y3 و قطر Dim
- متغیر Dim در برنامه ثابت هست اما چون دایره قصد حرکت دارد هر بار که متد move صدا زده شود مقدار متغیرهای x3 و y3 تغییر می کند در نتیجه سبب جابه جایی و حرکت دایره ما می شود. جلوتر این مفهوم را بهتر درک می کنید.

```
public void addNotify() {
          super.addNotify();
          animator = new Thread(this);
          animator.start();
     }
```

- متد addNotify() بعد از درج کنترل JPanel در JFrame فراخوانی می شود. وقتی این متد فراخوانی شود Thread برنامه ایجاد شده و استارت زده می شود.
- دوستان در ساخت یک انیمیشن و بازی که نیاز به چندنخی داریم بصورت ثابت از این متد استفاده می کنیم.

```
public void run() {

          while (true) {

               repaint();
               move();
               try {
                    Thread.sleep(10);
               } catch (InterruptedException e) {

                    e.printStackTrace();
               }

          }
     }
```

- همان طور که می دانید بعد از پیاده سازی اینترفیس Runnable پیاده سازی متد run در کلاس خود الزامی است.
- کل انیمیشن خود را یک نخ یا Thread قرار داده ایم و با اجرای برنامه متد run تنها یکبار اجرا می شود. برای این که چندین بار شکل دایره ای ما نمایش داده شود و حرکت پی در پی صورت بگیرد نیاز به یک حلقه بی نهایت داریم که متد های move و repaint چندین بار صدا زده شوند.
- برای کنترل سرعت انیمیشن خود از دستور Thread.sleep(10); استفاده می کنیم. چون ممکنه هنگام اجرای برنامه استثنایی رخ دهد دستور مذکور را درون بلوک try-catch قرار داده ایم.
- repaint(); همان طور که از اسمش مشخصه دستورات درون متد paint را با هربار صدا زده شدن اجرا می کند.

- ما نمیتوانیم برای اجرای دستورات درون متد paint ، مستقیم متد paint را صدا بزنیم برای اینکار از دستور repaint استفاده می کنیم.
- بار هر بار صدا زده شدن متد move() مختصات شکل دایره ای ما تغییر می کند و به همین دلیل عمل حرکت کردن اتفاق می افتد.

بررسی کلاس Main :

کلاس Main به دلیل این که از دستور جدیدی درونش استفاده نکردیم دیگه نیاز به توضیح ندارد.

تا اینجا.....

- با مفاهیم پایه گرافیک در جاوا
- چگونگی رسم اشکال هندسی
- حرکت اشکال در صفحه
- کاربرد Thread در ساخت انیمیشن
- حرکت شی و اشکال در صفحه
- و برخی مفاهیم دیگر آشنا شدید.

اکنون به ساخت یک بازی گرافیکی در جاوا نزدیک تر شده‌ایم. هر بازی نوعی انیمیشن است که با یک سری دستورات از جمله دستورات کنترلی صفحه کلید و موس استفاده از محدودیت‌ها و شروط و قوانینی که بازی مورد نظر ما دارا می باشد پیاده‌سازی می شود.

مثلا همین انیمیشن دایره که بررسی کردیم ، دایره از یک گوشه بصورت مورب شروع به حرکت می کند و به سمت پایین می‌رود و از داخل فریم با توجه به افزایش مختصات دایره(x,y)محو می شود، می توانیم با اعمال محدویت هایی خارج شدن دایره از صفحه را کنترل کنیم. مثلا می توان دستوری نوشت که اگر مختصات دایره به نقطه انتهایی فریم رسید x,y ها را به جای جمع زدن با یک، از یک کم کرد و سایر شروطی نظیر این که باعث شود که دایره از فریم خارج نشده و با دستوراتی که در گام های بعدی می‌خوانید حرکت دایره را در کنترل خود بگیریم و با کلیدهای جهتی صفحه کلید جهت حرکت دایره را تغییر دهیم. چگونکی کنترل انیمیشن(بازی) با صفحه کلید و سایر مفاهیم را در گام‌های بعدی با هم بررسی می‌کنیم.

گام چهارم: در قاب پنجره بمان

تا اینجا با گرافیک جاوا در ساخت فریم، پنل، رسم اشکال بر روی پنل و حرکت اشکال آشنا شدید. در برنامه قبل حرکت دایره در صفحه را داشتیم که بدون این که ما کنترلی روی آن داشته باشیم از یک گوشه ظاهر می‌شد و بصورت مایل حرکت می‌کرد و در گوشه دیگر محو می‌شد. در واقع دایره ما محو نمی‌شود، بلکه چون که مختصات x,y دایره رو به افزایش هست، وقتی که مختصات دایره از طول و عرض فریم بیشتر شد دیگه ما قادر به دیدن دایره در صفحه نخواهیم بود!

در این بخش می‌خواهیم جلوی محو شدن دایره در صفحه را بگیریم یعنی وقتی دایره از یک نقطه شروع به حرکت کرد و مثلا به چپ و راست یا پایین و بالای فریم برخورد کرد بجای عبور از از فریم کاری کنیم که برگردد. مثلا دایره را توپی فرض کنید که وقتی به دیوار برخورد می کند بجای عبور از دیوار ، توپ برگردد (قانون سوم نیوتن!)

در نگاه اول شاید دشوار بنظر برسد اما شما کمی صبر کنید، هر چیزی در نگاه اول سخت بنظر میرسه! ما می خواهیم که دایره از صفحه بیرون نره و هر بار که به دیواره صفحه(فریم) برخورد کرد قانون سوم نیوتن پیاده شه. برای اینکار اولین چیزی که به ذهن برنامه‌نویس خطور می کنه، کار با مختصات صفحه(فریم) و دایره است. یعنی باید طول و عرض فریم را بدانیم و کاری کنیم که x,y (مختصات) دایره بیشتر از طول و عرض فریم نشود. در ادامه خواهید دید که چگونه با چند دستور شرطی کاری می کنیم که دایره از صفحه عبور نکند.

فرض کنیم طول و عرض فریم شما به ترتیب 100 و 200 باشد و مختصات دایره شما یعنی x,y آن 0 و 0 باشد. وقتی دایره حرکت می کند به x,y های آن یک واحد اضافه می شود و همین طور افزایش می یابند تا اینکه x یا y یا هر دو دایره با طول و عرض فریم مساوی شوند. با داشتن طول و عرض فریم، در حالت عادی و بدون شرط همان طور که در برنامه قبل داشتیم دایره از فریم خارج می شود اما ما می خواهیم که از صفحه خارج نشود و بعد از برخورد با دیواره فریم برگشت داده شود. برای اینکه دایره از فریم عبور نکند و برگشت داده شود و فرض بر اینکه دایره به پایین یا سمت راست فریم برخورد کرده و برگشت داده می شود. می‌توان شرط زیر را برای آن نوشت:

```
if(x3+xa>main.getWidth()-Dim)
          xa=-1;

     if(y3+ya>main.getHeight()-Dim)
          ya=-1;
```

- در اینجا x3 و y3 مختصات قرار گیری دایره در صفحه فریم ما می باشد.
- xa و ya مقداری ثابتی هستند که با مختصات x و y دایره جمع می شوند.

- `main.getWidth()` عرض فریم را برای ما برمی گرداند.
- `main.getHeight()` طول یا ارتفاع فریم را برمیگرداند.
- `Dim` قطر دایره ما می باشد.

```
if(x3+xa>main.getWidth()-Dim)
```

- در این دستور گفته شده است اگرمختصات نقطه x (افقی) دایره بزرگ تر از عرض صفحه فریم شد مقدار xa را منفی یک (-۱) کن.چرا؟ چون با این کار باعث می شود مقدار ثابت xa که برابر +۱ بود به -۱ تغییر مقدار پیدا کند و باعث شود روند افزایشی مقدار x به روند کاهشی تغییر کند و دایره بعد از برخورد به دیواره سمت راست فریم به جای عبور برگردد مانند توپی که به دیواره سمت راست میخورد و برمیگردد.
- ما باید از همه لحاظ محدودیت ایجاد کنیم که دایره یا هر شکلی که داریم در محدوده فریم باقی بماند مثلا در شرط بالا نقطه y دایره از خط قرمز ارتفاع فریم عبور نکرده و هنوز کمتر از ارتفاع فریم می باشد اما ممکن است مقدار x دایره از خط قرمز محدوده سمت راست فریم عبور کند که با تدبیری که اندیشیدیم هیچگاه دایره ما از سمت راست فریم خارج نخواهد شد. برای سایر سمت های فریم نیز باید دستورات مربوطه را برای ایجاد محدودیت پیاده سازی کنیم.

```
if(y3+ya>main.getHeight()-Dim)
            ya=-1;
```

- این نیز برای ایجاد محدودیت در پایین صفحه فریم ما می باشد.
- به این دلیل منهای Dim کردهایم که وقتی نوک دایره به دیواره برخورد کرد همانجا برگشت بدهد. توجه کنید که اگر از قطر دایره کم نکنیم کل دایره در دیوار فرو می رود و بعد برگشت داده می شود.

اگه در برنامه قبل نگاه کرده باشید در حالت عادی حرکت دایره xa و ya دایره ۱ هست ما شرطی قرار دادیم که اگر به بدنه فریم برخورد کرد برگشت بدهد. یعنی بجای افزودن یک واحد به آن، یک واحد از آن کم کند. پس باید xa , ya برابر با 1- شوند. حالا برای بقیه حالت ها یعنی به دیواره چپ و بالا هم به همین روال عمل می‌کنیم. مثلا دستور دیواره بالا و چپ بصورت زیر است:

```
if(x3+xa<0)
      xa=1;

if(y3+ya<0)
      ya=1;
```

- در دستور زیر گفتیم اگه مختصات نقطه x دایره کمتر از ۰ یا عرض صفحه فریم بود روند کاهشی نقطه x را به روند افزایشی تغییر بده.

```
if(x3+xa<0)
	xa=1;
```

برای دستور زیر نیز گفته ایم که اگر مختصات نقطه y دایره کمتر از ۰ یا ارتفاع صفحه فریم شد روند کاهشی مختصات نقطه y دایره را به روند افزایشی تغییر بده.

حال کد برنامه قبلی را بصورت زیر تغییر می دهیم:

کلاس Board :

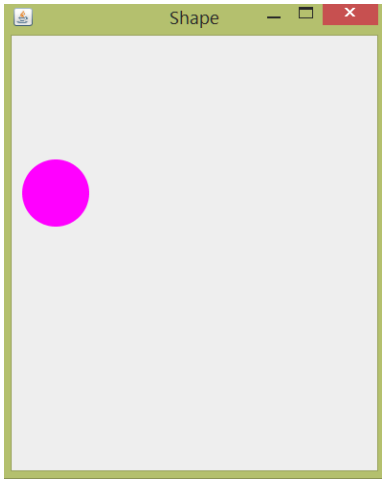
```
package javalikeGameTutorial;

import javax.swing.*;
import java.awt.*;

public class Board extends JPanel implements Runnable {

	int x3 = 0, y3 = 0, Dim = 70;
	int xa = 1, ya = 1;
	JFrame main;
	private Thread animator;

	public Board(JFrame main) {
		this.main = main;
	}

	public void move() {

		if (x3 + xa > main.getWidth() - Dim)
			xa = -1;

		if (y3 + ya > main.getHeight() - Dim)
			ya = -1;

		if (x3 + xa < 0)
			xa = 1;

		if (y3 + ya < 0)
			ya = 1;

		x3 = x3 + xa;
		y3 = y3 + ya;
```

```
	}

	public void paint(Graphics g) {
		super.paint(g);
		Graphics2D g2 = (Graphics2D) g;

		g2.setRenderingHint(RenderingHints.KEY_ANTIALIASING,
				RenderingHints.VALUE_ANTIALIAS_ON);

		g2.setColor(Color.magenta);
		g2.fillOval(x3, y3, Dim, Dim);

	}

	public void addNotify() {
		super.addNotify();
		animator = new Thread(this);
		animator.start();
	}

	public void run() {

		while (true) {

			repaint();
			move();
			try {
				Thread.sleep(10);
			} catch (InterruptedException e) {

				e.printStackTrace();
			}

		}

	}

}
```

و کلاس Main را بصورت زیر تغییر می دهیم:

```
package javalikeGameTutorial;

import javax.swing.JFrame;

public class Main extends JFrame {
	public Main() {
		add(new Board(this));
		setDefaultCloseOperation(JFrame.EXIT_ON_CLOSE);
		setSize(400, 500);
		setLocationRelativeTo(null);
		setTitle("Shape");
		setVisible(true);

	}

	public static void main(String[] args) {
		new Main();
	}
}
```

خروجی: بعد از run کردن کلاس Main خروجی بصورت تصویر(۶) خواهد بود:



تصویر(۶)

- نکته: دایره صورتی رنگ درحال حرکت می باشد و بعد از برخورد با دیواره های فریم برگشت میخورد.(دیگه اینجا پی دی اف هست باید خودتون کدو اجرا کنید تا حرکت رو ببینید :-))

توضیحات کلاس Board :

دستوراتی که جدید و غیر تکراری هست را بررسی می کنیم.

```
JFrame main;
```

- یک شی از نوع کلاس JFrame تعریف کرده ایم.

```
•       public Board(JFrame main) {
•            this.main = main;
•       }
```

- در سازنده این کلاس یک پارامتر از نوع کلاس JFrame تعریف کرده و پارامتر سازنده را درون main که در بدنه کلاس تعریف کردیم می ریزیم.

```
public void move() {

        if (x3 + xa > main.getWidth() - Dim)
            xa = -1;

        if (y3 + ya > main.getHeight() - Dim)
            ya = -1;

        if (x3 + xa < 0)
            xa = 1;

        if (y3 + ya < 0)
            ya = 1;

        x3 = x3 + xa;
        y3 = y3 + ya;

    }
```

- عمده تغییرات ما در متد move موجود در کلاس Board می باشد.
- در این متد شروطی که موجب محدودیت و حرکت دایره در چارچوب فریم می شود را ایجاد کرده ایم.

```
x3 = x3 + xa;
y3 = y3 + ya;
```

این دو دستور کار حرکت دایره ما را انجام می دهند. و تنها دستورات زیر است که نحوه و جهت حرکت را تعیین می کنند:

```
if (x3 + xa > main.getWidth() - Dim)
            xa = -1;
```

```
        if (y3 + ya > main.getHeight() - Dim)
            ya = -1;

        if (x3 + xa < 0)
            xa = 1;

        if (y3 + ya < 0)
            ya = 1;
```

- شما می توانید با دستکاری شروط فوق جهت حرکت دایره را با توجه به نیاز تغییر بدهید.
- دلیل که شی main رو ابتدای کلاس تعریف کردیم این بود که به متدهای JFrame برنامه که در اینجا متدهایی برای به دست آوردن طول و عرض فریم برنامه می باشد دسترسی پیدا کنیم.
- سایر دستورات کلاس Board بدون تغییر می باشد.

بررسی کلاس Main :

```
public class Main extends JFrame {
```

همان طور که مشاهده می کنید کلاس Main کلاس JFrame را به ارث برده است ، پس کلاس Main فرزند کلاس JFrame هستش و به عبارتی خود یک JFrame محسوب می شود.

```
add(new Board(this));
```

- با متد add شی از کلاس JPanel را به JFrame خود که اضاف کرده ایم.
- کلمه کلیدی this اشاره به شی کلاسی که درونش هستیم می کند یعنی شی کلاس Main و از طرفی پارامتر سازنده کلاس Board از نوع کلاس JFrame هستش به همین خاطر کلمه کلیدی this را جایگزین پارامتر سازنده کلاس Board کرده ایم.
- سایر دستورات را قبلا(ابتدای همین جلسه) توضیح داده ایم.

تا الان با حرکت اشکال(انیمیشن) و لحاظ کردن محدودیت بر حرکت اشکال آشنا شدید. در بخش بعدی با چگونگی کنترل اشکال با صفحه کلید آشنا خواهیم شد.

گام پنجم: تحت فرمان من

در این بخش با چگونگی اعمال کنترل بر دایره موجود در قصه ما با صفحه کلید آشنا خواهید شد. منظور از کنترل بر دایره یعنی این که مثلا با کلید های جهتی کیبورد، جهت حرکت دایره را تغییر دهیم. یا این که با زدن دکمه Delete دایره را کلا از صفحه حذف کنیم، و یا بازدن دکمه Space حرکت دایره متوقف شود و....

اینترفیس KeyListener:

ما از اینترفیس KeyListener برای استفاده از دکمه های کیبورد استفاده می کنیم.این اینترفیس رخدادهای مربوط به دکمه های کیبورد را تشخیص می دهد و با توجه به متدهایی که دارد واکنش مربوطه را انجام می دهد . این اینترفیس دارای سه متد زیر می باشد:

```
public abstract void keyPressed(KeyEvent e);
```

- وقتی دکمه کیبورد را فشار می دهیم این متد صدا زده می شود و دستورات درون بدنه آن اجرا می شود.

```
public abstract void keyReleased(KeyEvent e);
```

- وقتی دکمه کیبورد را رها می کنیم این متد صدا زده می شود و دستورات درون بدنه آن اجرا می شود.

```
public abstract void keyTyped(KeyEvent e);
```

- وقتی شما درحال فشردن یکی از دکمه های کیبورد هستید(تایپ می‌کنید) این متد صدا زده شده و دستورات درون آن اجرا می شود.

همه متدهای بالا پارامتری با نام e که از نوع کلاس KeyEvent می باشد را دارا می باشند. هر کلید یا دکمه کیبورد یک KeyEvent می باشد. برای هر دکمه کیبورد یک مقدار ثابتی درنظر گرفته شده است که با زدن دکمه مربوطه اون مقدار درون پارامتر e که از نوع کلاس KeyEvent می باشد ریخته می شود. به زبان ساده با زدن هر کدام از دکمه های کیبورد متدهای مربوط به اینترفیس KeyListener صدا زده می شود و مقدار ثابت اون دکمه درون متغیر e که از نوع کلاس KeyEvent هستش ریخته می شود. حالا چطور تشخیص دهیم که این رویداد رخ داده مربوط به کدام دکمه کیبورد هستش؟! صبر داشته باشید جلوتر بررسی می کنیم :-)

نکته :

- برای استفاده از کیبورد در جاوا باید کلاس خود را impelements به اینترفیس KeyListener راکنیم.
- برای استفاده از اینترفیس KeyListener باید پکیج زیر را در کلاس خود import کنید:

```
import java.awt.event.KeyListener;
```

- برای استفاده از کلاس KeyEvent باید پکیج زیر را در برنامه خود import کنید:

```
import java.awt.event.KeyEvent;
```

- البته IDE های ایکلیپس یا نتبینز خود در صورت import نکردن پکیج ها بهتون اخطار میده و بهتون پکیج مورد نظر را پیشنهاد می دهد.

خب در ادامه مثال می زنیم و و از روی مثال توضیح می دهیم اینجوری بهتره :-)

قصد داریم همان برنامه قبل خود را توسعه دهیم. تا اینجا ما یک دایره صورتی رنگ رسم کردیم و آن را به حرکت در آوردیم و با اعمال محدودیت دایره را در چارچوب صفحه فریم خود نگه داشتیم. حالا قصد داریم با کلید های جهتی کیبورد جهت حرکت دایره صورتی خود را تغییر دهیم به نوعی فرمان بهش بدیم و در کنترل خود دربیاریم.

کلاس Board :

```
package javalikeGameTutorial;

import javax.swing.*;

import java.awt.*;
import java.awt.event.KeyEvent;
import java.awt.event.KeyListener;

public class Board extends JPanel implements Runnable, KeyListener {
	int x3 = 0, y3 = 0, Dim = 70;
	static int xa = 1, ya = 1;
	JFrame main;
	private Thread animator;

	public Board(JFrame main) {
		this.main = main;
	}
```

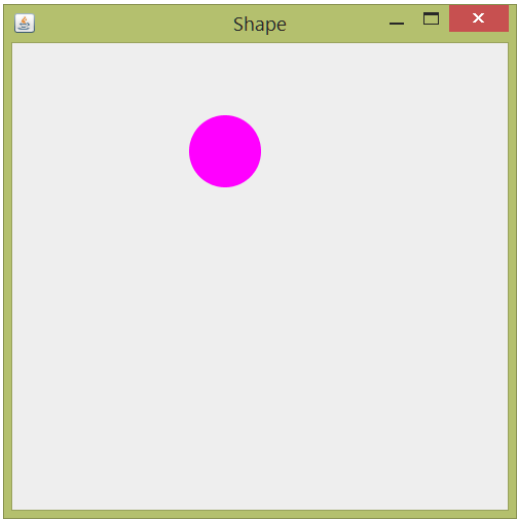
```
	public void move() {
		if (x3 + xa > main.getWidth() - Dim)
			xa = -1;
		if (y3 + ya > main.getHeight() - Dim)
			ya = -1;
		if (x3 + xa < 0)
			xa = 1;
		if (y3 + ya < 0)
			ya = 1;
		x3 = x3 + xa;
		y3 = y3 + ya;

	}

	public void paint(Graphics g) {
		super.paint(g);
		Graphics2D g2 = (Graphics2D) g;
		g2.setRenderingHint(RenderingHints.KEY_ANTIALIASING,
				RenderingHints.VALUE_ANTIALIAS_ON);
		g2.setColor(Color.magenta);
		g2.fillOval(x3, y3, Dim, Dim);

	}

	public void addNotify() {
		super.addNotify();
		animator = new Thread(this);
		animator.start();
	}
	public void run() {
		while (true) {
			repaint();
			move();
			try {
				Thread.sleep(10);
			} catch (InterruptedException e) {

				e.printStackTrace();
			}
		}
	}
	@Override
	public void keyPressed(KeyEvent e) {
		if (e.getKeyCode() == KeyEvent.VK_RIGHT)
			xa = 1;
		if (e.getKeyCode() == KeyEvent.VK_LEFT)
```

```
                xa = -1;
            if (e.getKeyCode() == KeyEvent.VK_UP)
                ya = -1;
            if (e.getKeyCode() == KeyEvent.VK_DOWN)
                ya = 1;

        }
        @Override
        public void keyReleased(KeyEvent e) {
            // TODO Auto-generated method stub
        }
        @Override
        public void keyTyped(KeyEvent e) {
            // TODO Auto-generated method stub
        }
}
```

کلاس Main :

```
package javalikeGameTutorial;

import javax.swing.JFrame;

public class Main extends JFrame {

    public Main() {
        add(new Board(this));
        Board bord = new Board(this);
        addKeyListener(bord);
        setDefaultCloseOperation(JFrame.EXIT_ON_CLOSE);
        setSize(500, 500);
        setLocationRelativeTo(null);
        setTitle("Shape");
        setVisible(true);

    }

    public static void main(String[] args) {
        new Main();
    }
}
```

خروجی: تصویر(۷) خروجی برنامه یک دایره صورتی رنگ متحرک هست که ما می توانیم با دکمه های جهتی کیبورد جهت حرکت دایره را تغییر دهیم. خب اینجا آموزش در قالب پی دی اف هست و نمی تونیم نحوه کار کردن برنامه رو نمایش دهیم دیگه خودتون زحمت اجرا کردن برنامه را بکشید و با چشم خود ببینید ☺



تصویر(۷)

توضیحات مربوط به کلاس Board : تنها دستوراتی که جدید هستند را بررسی کرده ایم.

```
import java.awt.event.KeyEvent;
import java.awt.event.KeyListener;
```

- چون قصد داریم در برنامه خود از دکمه های کیبورد استفاده کنیم پکیج های های مربوطه را import می کنیم.

```
public class Board extends JPanel implements Runnable, KeyListener {
```

- خب برای تشخیص دکمه های کیبورد در برنامه خود کلاس خود را implements به اینترفیس KeyListener کرده ایم.

```
    public void keyPressed(KeyEvent e) {
        if (e.getKeyCode() == KeyEvent.VK_RIGHT)
            xa = 1;
        if (e.getKeyCode() == KeyEvent.VK_LEFT)
            xa = -1;
        if (e.getKeyCode() == KeyEvent.VK_UP)
            ya = -1;
        if (e.getKeyCode() == KeyEvent.VK_DOWN)
            ya = 1;
```

```
	}
	@Override
	public void keyReleased(KeyEvent e) {
		// TODO Auto-generated method stub
	}
	@Override
	public void keyTyped(KeyEvent e) {
		// TODO Auto-generated method stub
	}
```

- وقتی کلاس خود را implements به اینترفیس KeyListener کنیم به صورت خودکار سه متد مربوطه این اینترفیس در کلاس ما پیاده سازی می شود.
- ما تنها دستورات خود را درون بدنه متد keyPressed قرار داده ایم. این کار باعث می شود وقتی دکمه ای از کیبورد را فشار می دهیم دستورات درون بدنه این متد اجرا می شود.

```
if (e.getKeyCode() == KeyEvent.VK_RIGHT)
			xa = 1;
		if (e.getKeyCode() == KeyEvent.VK_LEFT)
			xa = -1;
		if (e.getKeyCode() == KeyEvent.VK_UP)
			ya = -1;
		if (e.getKeyCode() == KeyEvent.VK_DOWN)
			ya = 1;
```

- خب ما در بدنه متد keyPressed شرط های فوق را قرار دادیم.

```
e.getKeyCode()
```

- این دستور کلید یا دکمه از کیبورد را که انتخاب کرده ایم را برای ما برمی گرداند.
- همان طور که از قبل گفتیم هر دکمه کیبورد یک مقدار ثابت و منحصر به فردی دارد که وقتی انتخاب می شود مقدار آن درون متغیر e که از نوع کلاس KeyEvent هست ریخته می شود. حالا برای این که بررسی کنیم که این رویداد رخ داده کیبورد مربوط به کدام دکمه از صفحه کلید است از دستور e.getKeyCode() استفاده می کنیم.

```
KeyEvent.VK_RIGHT
```

- مقدار ثابت دکمه جهتی سمت راست کیبورد می باشد.

```
KeyEvent.VK_LEFT
```

- مقدار ثابت دکمه جهتی سمت چپ کیبورد می باشد.

KeyEvent.***VK_UP***

- مقدار ثابت دکمه جهتی سمت بالا کیبورد می باشد.

KeyEvent.***VK_DOWN***

- مقدار ثابت دکمه جهتی سمت پایین کیبورد می باشد.
- پس هر دکمه کیبورد مقدار ثابت مربوط به خود را دارا می باشد.

```
if (e.getKeyCode() == KeyEvent.VK_RIGHT)
                    xa = 1;
```

- در این شرط گفته شده اگر رویداد رخ داده از کیبورد برابر دکمه جهتی سمت راست کیبورد بود مقدار xa را برابر عدد ۱ قرار بده.
- همان طور که می دانید برای حرکت دایره به سمت راست صفحه فریم باید به مختصات نقطه x دایره افزوده شود به همین خاطر xa را برابر +۱ قرار داده ایم.

```
if (e.getKeyCode() == KeyEvent.VK_LEFT)
                    xa = -1;
```

- در این شرط گفته شده اگر رویداد رخ داده از کیبورد برابر دکمه جهتی سمت چپ کیبورد بود مقدار xa را برابر عدد -۱ قرار بده.
- برای حرکت دایره به سمت چپ صفحه فریم باید از مختصات نقطه x دایره کم شود به همین خاطر xa را برابر -۱ قرار داده ایم.
- محور x های فریم را در نظر بگیرید مثلا بین ۰ تا ۱۰۰ هست و دایره ما در نقطه ۵۰ قرار گرفته است حالا برای این که دایره به نقطه ۱۰۰ برود باید بر مقدار x دایره اضاف شود یا اگه قصد داریم دایره به سمت ۰ برود باید از مقدار x دایره کم شود.

```
if (e.getKeyCode() == KeyEvent.VK_UP)
                    ya = -1;
```

- در این شرط گفته شده اگر رویداد رخ داده از کیبورد برابر دکمه جهتی سمت بالا کیبورد بود مقدار ya را برابر عدد -۱ قرار بده.
- برای حرکت دایره به سمت بالا صفحه فریم باید از مختصات نقطه y دایره کم شود به همین خاطر ya را برابر -۱ قرار داده ایم.

- محور y های فریم را در نظر بگیرید مثلا بین ۰ تا ۱۰۰ هست و دایره ما در نقطه ۵۰ قرار گرفته است حالا برای این که دایره به نقطه ۱۰۰ برود باید بر مقدار y دایره اضاف شود یا اگه قصد داریم دایره به سمت ۰ برود باید از مقدار y دایره کم شود.

```
if (e.getKeyCode() == KeyEvent.VK_DOWN)
            ya = 1;
```

- در این شرط گفته شده اگر رویداد رخ داده از کیبورد برابر دکمه جهتی سمت پایین کیبورد بود مقدار ya را برابر عدد ۱+ قرار بده.
- برای حرکت دایره به سمت پایین صفحه فریم باید به مختصات نقطه y دایره افزوده شود به همین خاطر ya را برابر ۱+ قرار داده ایم.
- دوستان قالب شرط ها برای بررسی دکمه های کیبورد به همین صورت می باشد و تنها کافیه مقدار ثابت دکمه کیبورد مورد نظر و هدفتون از عملکرد دکمه کیبورد مورد نظر را برای خودتون اعمال کنید.
- سایر دستورات را در همین جلسه آموزشی بررسی کرده ایم.

بررسی کلاس Main :

```
Board bord = new Board(this);
        addKeyListener(bord);
```

- برای افزودن KeyListener به یک component (اجزای گرافیکی) از متد addKeyListener استفاده می کنیم.
- در اینجا KeyListener ما کلاس Board می باشد. پس از کلاس Board شی ایجاد کرده و شی ایجاد شده را به عنوان پارامتر به متد addKeyListener داده ایم.
- در اینجا متد addKeyListener مربوط به کلاس Main هست. از طرفی کلاس Main به دلیل این که کلاس JFrame را به ارث برده است خود یک JFrame و component حساب می شود به همین دلیل می تواند متد addKeyListener را صدا بزند.
- چرا متد addKeyListener را در کلاس Main صدا زدیم؟ چون قصد داریم KeyListener (شونده رویداد کیبورد) را به JFram برنامه خود اضاف کنیم. از طرفی کلاس Main کلاس JFrame را به ارث برده است ، پس کلاس Main خود یک JFram حساب می شود که صلاحیت صدا زدن متد addKeyListener را دارا می باشد. با این کار هر رویداد کیبوردی که روی فریم برنامه رخ دهد تشخیص داده می شود.

- در کل هر اجزای گرافیکی (component) در جاوا که متد addKeyListener صدا بزند باعث می شود KeyListener (شونده رویداد کیبورد) به ان اجزای گرافیکی اضاف شود. مثلا اگر به جای JFrame یک اجزای گرافیکی مثل JTeaxtArea متد addKeyListener را صدا بزند هر بار که در textarea در حال نوشتن با کیبورد هستیم متدهای اینترفیس KeyListener صدا زده می شوند.

نتیجه گیری: هر وقت در کلاس خود اینترفیس KeyListener را پیاده سازی کردیم باید با متد addKeyListener کلاسی که اینترفیس KeyListener را implements کرده است را به اجزای گرافیکی مورد نظر اضاف کنیم.

احتمالا توضیحات گیج کننده باشد!! اصلا نگران نشید این یک اتفاق عادی است که برای هر کسی ممکنه بیفته! کافی است به دستورات و کدهای برنامه چندین بار نگاه کنید و برای خودتون تغییرش بدید و مثال های بیشتری رو ببینید ، یواش یواش مفهومش براتون روشن می شود.

بیایید با هم مرور کنیم تا اینجا با چه چیزهایی آشنا شدیم:

۱. با محیطی که قرار بود بازی در آن اجرا شود آشنا شدیم.
۲. با انواع کلاس هایی که import می کردیم تا از متدها و ویژگی هاشان استفاده کنیم آشنا شدیم.
۳. با چگونگی رسم اشکال و متدهایی که با استفاده از آن شکل های هندسی را رسم میکردیم آشنا شدیم.
۴. با حرکت اشکال (انیمیشن) آشنا شدیم.
۵. با ایجاد محدودیت و چارچوب برای حرکت اشکال خود آشنا شدیم.
۶. و با کنترل حرکت اشکال با صفحه کلید آشنا شدیم.

دیگر برای ساخت یک بازی دوبعدی به چه چیزی نیاز داریم؟ با بکارگیری موارد ۱ تا ۶ قوانین بازی مورد نظرمان و کمی خلاقیت می توانید هر بازی دوبعدی که بسازید.

- کلا یک بازی دوبعدی از تعداد اشکال و اشیا که کاراکترهای بازی رو تشکیل می دهند، حرکت و انیمیشن ، چارچوب محیط حرکت و فرمان دادن با صفحه کلید یا موس ایجاد می شود که ما با مثال ساده با طراحی یک دایره صورتی رنگ و به حرکت درآوردن آن در یک چارچوب مشخص و کنترل آن با کلید های کیبورد این مفاهیم را آموزش دادیم.

- ما در گام دوم حل مسئله بازی گرفتن تخم مرغ ابتدا مفاهیم مشترک و عمومی که در هر بازی دوبعدی در جاوا مورد نیاز است را بررسی کردیم.
- در جلسه بعدی روش ساخت بازی دوبعدی گرفتن تخم مرغ ها را به زبان جاوا به کمک همین مفاهیمی که در این جلسه بررسی کردیم به صورت پروژه محور آموزش خواهیم داد. امیدوارم این مفاهیم رو به صورت واضح توضیح داده باشم.

پیروز و موفق باشید

هر روز مفاهیم و مثال های جدید به سایت اضافه می شود برای اطلاع از مطالب جدید روی سایت عضو کانال شوید.