جزوه جهت آموزش مجازی

برنامه نویسی موبایل

جلسه ۱

مدرس:

زینب نادری

مروری بر جلسه گذشته

در جلسه اول که در دانشگاه برگزار شد، زبان جاوا را معرفی و دلیل انتخاب این نام را شرح دادیم. با **انواع داده ها، نحوه تعریف متغییر** آشنا شدیم که این نحوه تعریف، تفاوتی با زبان سی شارپ نداشته است. همچنین **انواع عملگرهای** موجود بیان شد: عملگرهای محاسباتی، رابطه ای، منطقی، بیتی و ترکیبی(از ترکیب عملگرهای محاسباتی و علامت = ایجاد می شود). سپس **دستورات شرطی** (شامل دستورات if ، if-else ، if-else if) توضیح داده شد. دستور Switch که همانند زبان سی شارپ در این زبان نیز مورد استفاده قرار می گیرد. در انتها نحوه کار با **حلقه ها** (حلقه while، do while و for) بیان شد.

بخشی از سرفصل درس:

۱. معرفی زبان جاوا وآشنایی با انوع داده ها، تعریف متغییرها، عملگرها و دستورات شرطی، حلقه ها و توابع در جاوا

۲. شی گرایی و وراثت در جاوا

۳. نصب و راه اندازی اندروید استدیو و شبیه ساز آن

۴. آشنایی با محیط اندروید استدیو و تولید اولین پروژه اندرویدی

۵. آشنایی با چینش عناصر در اندروید و Text View

۶. آشنایی با رنگ ها و ابزار Button و Event

۷. آشنایی با چرخه حیات Activityدر اندروید

۸. آشنایی با انواع منو در اندروید و روش های تعریف منو

نکته: در جلسه حضوری تقریبا قسمت زیادی از بخش ۱ سرفصل آموزش داده شد، در نظر داریم در طی ۵ جلسه مجازی سایر قسمتها را بیان کنیم. تا در سال جدید با حجم زیادی از مطالب ناگفته روبرو نشویم.

۱.قالب کلی برنامه در زبان جاوا

به ساختار کلی یک برنامه در جاوا دقت کنید.

```
public class نام کلاس
{
    public static void main(String[] args)
    {
        // دستورات و کدهای برنامه
    }
}
```

خط اول برنامه ما تعریف یک کلاس است. کلاس‌ها اجزای اصلی برنامه نویسی شیء‌گرا هستند. در برنامه نویسی شیء‌گرا هر چیزی یک شیء است و از آن جا که اشیاء را در جاوا با کلاس‌ها پیاده‌سازی می‌کنند، پس هر چیزی که در برنامه جاوا داریم درون کلاس‌ها قرار می‌گیرند(در بخش شی گرایی بیشتر توضیح می دهیم).

بهتر است بدانیم حرف اول نام کلاس همیشه باید با یک حرف بزرگ شروع شود چرا که زبان جاوا نسبت به بزرگی و کوچکی حروف حساس است. نکته دیگری که می بایست به آن توجه نمود این است که نام فایل جاوا باید همنام با نام کلاس باشد. از این رو هنگام ذخیره فایل، لازم است که آن را با استفاده از نام کلاس ذخیره کنید و فرمت جاوا را به انتهای نام فایل اضافه کنید.(یعنی اگر نام کلاس شما Test1 باشد، نام فایل ذخیره ای شما چیست؟ بله، درسته Test1.java)

بدنه کلاس درون { و } قرار می‌گیرد. خط دوم تعریف یک متد است. البته متد main() یک متد خاص و ویژه است. بدنه متد هم مانند کلاس درون { و } قرار می‌گیرد. در داخل هر متد چندین خط دستورات داریم.

۱.۱. متدmain

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

این متد، متد اصلی کلاس جاوا است و زمانی که یک برنامه جاوا شروع به راه اندازی می شود کلاس شروع شونده به دنبال اجرای متد اصلی خود یعنی متد main می باشد . اگر این متد در داخل کلاس وجود نداشته باشد خطایی با این مضمون رخ خواهد داد که متد اصلی برنامه موجود نیست. از این رو تمامی متدها و کدهایی که می خواهیم به محض راه اندازی برنامه شروع به اجرا شود باید داخل این متد گنجانده شود.

مثال:

```
public class FisrtProgram
{
    public static void main (String[] args)
    {
        System.out.println("first Program of Java!");
```

```
    }
}
```

در این مثال، خط اول یک کلاس تعریف شده، خط دوم تعریف متد main() است، برنامه بسیار ساده است و هیچ متد دیگری ندارد. در مثال ما در متد main() فقط یک خط دستور وجود دارد. این دستور عبارت first program of Java را در خروجی نمایش می‌دهد.

۲. دستور try-catch

اگر در برنامه ای که نوشته اید، خطایی رخ دهد چه اتفاقی می افتد؟ به مثال زیر دقت کنید.

```
public class Test1
{
    public static void  main (String args[])
    {
       int data=50/0;
       System.out.println("rest of the code...");
    }
}
```

خروجی:

```
Exception in thread main java.lang.ArithmeticException:/ by
zero
```

همان طور که مشاهده می کنید، بقیه ی دستورات اجرا نشده و متنی در خروجی چاپ نمی شود. در سایر برنامه ها ممکن است پس از خطی که خطا در آن رخ می دهد، ۱۰۰ دستور دیگر وجود داشته باشد که اگر خطا رخ دهد، تمامی آن ها اجرا نخواهند شد. زیرا به محض رخدادن خطا، اجرای برنامه متوقف می شود و خطا به کاربر نمایش داده می شود و ادامه برنامه اجرا نخواهد شد. حال اگر بخواهیم با بروز یک خطا، پیغامی مناسب به کاربر نمایش داده شود و ادامه کدها نیز اجرا شود چه باید بکنیم؟

می توان خطا را با استفاده از try…catch اداره کرد.

```
Try
{
کدی که احتمال بروز خطا در آن هست
}
catch(Exception_class_Name ref)
{
کدی که در صورت بروز خطا اجرا می شود
}
```

کدی که ممکن است اجرای آن منجر به خطا شود، بایستی داخل قطعه کد try قرار داده شده و در واقع در این قسمت از برنامه امتحان شود تا امکان اجرای موفقیت آمیز آن مشخص گردد . قطعه کد catch برای

مدیریت خطاهای رخ داده در بدنه یtry ، بکار می روند. این قطعه کد منحصرا بایستی پس از قطعه try درج شود. می توانید پس ازیک قطعه کد try چندین قطعه کد catch داشته باشیدکه هریک Exception_class_Name متفاوتی خواهدداشت و خطاهای مختلفی را چک می کند که برای هر خطا، پیغامی منحصربفرد نمایش دهد.

حال مثال بالا را با استفاده از قطعه کد try-catch نوشته و مشکل آن را برطرف می سازیم.

```
public class Test2
{
   public static void  main(String args[])
   {
      try
      {
         int data=50/0;
      }catch ()
      {
         System.out.println("خطایی رخ داده است، دقت کنید");
      }
      System.out.println("rest of the code...");
   }
}
```

خروجی:

```
خطایی رخ داده است، دقت کنید
rest of the code...
```

همان طور که در مثال بالا مشاهده می کنید، باقی کد اجرا شده و رشته ی مورد نظر در خروجی چاپ می شود .

۳. توابع در جاوا (java Function)

زمانی که کدها زیاد می شوند نظم و خوانایی برنامه کاهش می یابد برای حل این مشکل، آن مجموعه از کدها که یکسری روال مشخص را انجام می دهند را درون یک تابع قرار می دهیم و از بدنه اصلی برنامه آن تابع را صدا می زنیم.

۱.۳. تعریف یک تابع

```
public static int methodName(int a, int b) {
    // دستورات درون تابع
}
```

public static: نوع دسترسی تابع

int : نوع خروجی (مقدار برگشتی) تابع

نکته: در صورتی که بخواهیم تابع چیزی را برنگرداند از کلمه void استفاده میکنیم .

methodname : نام دلخواه برای تابع

int a,int b : پارامترهای تابع

مثال: در زیر یک تابع تعریف می کنیم که دو عدد می گیرد و کوچکترین آنها را برمی گرداند.

```
public  static  int  minFunction(int n1, int n2)
{
   int min;
   if (n1 > n2)
      min = n2;
   else
      min = n1;
   return min;
}
```

۲.۳.صدا زدن تابع

هنگامی که در برنامه بخواهیم از تابع استفاده کنیم ابتدا نام تابع و در صورت داشتن پارامترها ، آنها را در پرانتز گذاشته و به تابع ارسال می کنیم . در صورتی که تابع مقدار برگشتی نیز داشته باشد آن را به متغیر یا دستور دلخواه نسبت می دهیم.

مثال: در زیر نحوه صدا زدن تابع را خواهید دید :

```
public  class  Example_MinNumber
{
    public static void main(String[] args)
    {
       int a = 11;
       int b = 6;
       int c = minFunction(a, b);
       System.out.println("Minimum Value = " + c);
    }
    /** تابع **/
    public static int minFunction(int n1, int n2)
    {
       int min;
       if (n1 > n2)
          min = n2;
       else
          min = n1;
       return min;
    }
```

```
}
```

خروجی این برنامه بصورت زیر است:

```
Minimum value = 6
```

اگر دقت کنید مطابق قالب اصلی برنامه در جاوا، در مثال بالا، ابتدا یک کلاس به نام Example_MinNumber تعریف کردیم (نام فایل برنامه هم باید همنام با این نام باشد یعنی Example_MinNumber.java باشد) و سپس متد Main را تعریف کردیم که گفتیم، به محض اجرا شدن برنامه، ابتدا سراغ این متد می رود و دستورات درون آن را اجرا می کند. در خط چهارم از متد main دستوری قرار دارد که تابع minFunction را فراخوانی می کند و با رسیدن به این کد سراغ تعریف تابع می رود و کدهای درون تابع را اجرا می کند و بعد از اجرای تابع مجددا به برنامه اصلی بازمی گردد و خط آخر که دستور چاپ می باشد، را اجرا می کند.

تمرین ۱:

برنامه ای مطابق مثال بالا بنویسید که دو عدد را دریافت کند و با استفاده از یک تابع مجموع آن را در خروجی نمایش دهد.

راهنمایی: بایستی کلاسی تعریف کنید و درون کلاس یک متد main تعریف کرده که دو عدد درون آن تعریف می شود و با فراخوانی تابعی که برای جمع، تعریف کردید، مجموع آن را محاسبه و سپس چاپ کنید.