

## به نام خدا

### آموزش چهل و سوم

اهداف آموزشی این قسمت عبارتند از:

۱. به کارگیری از Constructor ها در برنامه نویسی
۲. به کارگیری از دستور this به همراه پارامترها
۳. استفاده از کلاس Math جاوا
۴. استفاده از متد round جهت رند کردن اعداد
۵. استفاده از متدهایی که داده ای را return می کنند

پس از آشنایی با مفهوم Constructor ها در قسمت چهل و یکم، به منظور درک عملی Constructor ها سناریویی تعریف کرده و به طور عملی این نوع از متدها را مورد استفاده قرار می دهیم. فرض کنیم که می خواهیم برنامه ای طراحی کنیم که در ابتدا نام ما را گرفته سپس نام خودروی خود را وارد برنامه کرده، در نهایت این برنامه به ما می گوید میزان مصرف خودروی ما در هر صد کیلومتر چند لیتر است.

برای شروع کار یک پروژه جدید تحت عنوان 43<sup>rd</sup> Session به معنی "جلسه چهل و سوم" ایجاد می کنیم و کلاسی به نام FuelConsumption به معنی "مصرف سوخت" در آن تعریف می کنیم. اکنون پیش از شروع هر کاری اقدام به ایجاد دو Field در کلاس خود تحت عناوین نام، نام خودرو می کنیم:

```
public class FuelConsumption {  
    String name;  
    String carName;  
}
```

همانطور که در کد فوق ملاحظه می شود Field اول که از جنس String است name به معنی "نام" نامیده شده است که قرار است نام صاحب خودرو را در خود ذخیره سازد. سپس یک Field دیگر از همین جنس تحت عنوان carName به معنی "نام خودرو" ایجاد کرده ایم که

دوره آموزش جاوا

کلیه حقوق متعلق به وب سایت نردبان است.

مدرس: بهزاد مرادی

نام خودروی ما را قرار است در خود ذخیره سازد. اکنون نیاز است تا یک Constructor تعریف کنیم که مقادیر اولیه این Field ها را مشخص سازد. برای این منظور کد فوق را به شکل زیر تکمیل می کنیم:

```
public class FuelConsumption {
    String name;
    String carName;

    // Constructor
    public FuelConsumption(String name, String carName) {
        this.name = name;
        this.carName = carName;
    }
}
```

همانطور که در کد فوق ملاحظه می شود یک Comment قرار داده ایم که بدانیم کد بعد از آن مربوط به Constructor است. سپس Constructor خود را به گونه ای تعریف کرده ایم که می بایست برای اجرا، دو پارامتر ورودی آن مشخص شوند. به عبارت دیگر مادامیکه کاربر نام و نام خودروی خود را وارد برنامه نکند، شیء ساخته شده از روی کلاس FuelConsumption کار نخواهد کرد. کاری که در این برنامه انجام داده ایم این است که نام پارامترهای Constructor خود را هم نام Field ها در نظر گرفته ایم (لازم به ذکر است که برای پارامترها هر نامی می توان در نظر گرفت اما از آنجا که می خواهیم تعداد نام های متغیرهای برنامه ما به حداقل برسد، این نام ها را هم نام با نام Field ها در نظر گرفته ایم).

زمانیکه نام پارامترها با نام Field ها یکی باشد، Compiler در حین اجرا مقادیر Field ها را مد نظر قرار خواهد داد. برای رفع این مشکل از کلید واژه this استفاده می کنیم. همانطور که در کد فوق ملاحظه می شود پیش از name از کلید واژه this به علاوه یک نقطه استفاده شده و در نهایت یک علامت مساوی قرار داده ایم و مجدد واژه name را نوشته و یک علامت ; پس از آن قرار داده ایم. به زبان ساده این بدان معنا است که "این" پارامتری که نام آن name است، مقدار آن برابر با Field یی است که نام آن name می باشد. در مورد carName هم شرایط به همین صورت است. از آنجا که نیاز است مقدار بازگشتی این Field ها بارها در برنامه ما مورد استفاده

قرار گیرند، از این رو دو متد با قابلیت return برای هر یک از Field ها به صورت زیر تعریف می کنیم:

```
public class FuelConsumption {
    String name;
    String carName;
    double fuelConsumption;

    // Constructor
    public FuelConsumption(String name, String carName) {
        this.name = name;
        this.carName = carName;
    }

    // Return Methods
    public String returnName() {
        return name;
    }

    public String returnCarName() {
        return carName;
    }
}
```

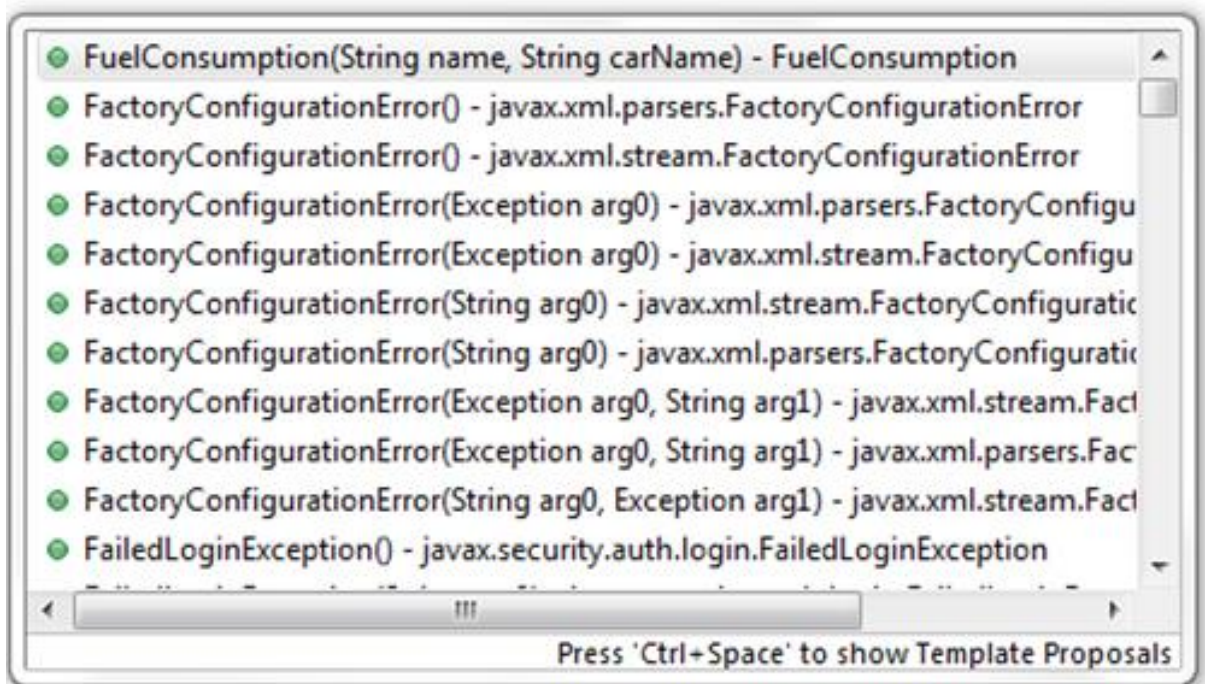
همانطور که در کد فوق ملاحظه می شود یک Comment با عبارت Return Methods در برنامه قرار داده و سپس سه متد از جنس return تعریف کرده ایم. متد اول که reurnName به معنی "مقدار نام را بازگردان" نام دارد قرار است مقدار Field قرار گرفته در کلاس تحت عنوان name را بازگرداند. متد دوم که returnCarName به معنی "مقدار متغیر نام خودرو را بازگردان" نام دارد قرار است مقدار Field قرار گرفته در این کلاس به نام carName را بازگرداند.

اکنون کار را در این مرحله برای این کلاس متوقف کرده و اقدام به ایجاد کلاسی جدید تحت عنوان ActionClass به معنی "کلاس عملیاتی" می کنیم. از آنجا که این کلاس قرار است به منزله نقطه شروع برنامه باشد می بایست در حین ساخت آن گزینه public static void main را برای آن تیک بزنیم. پس از ایجاد این کلاس اقدام به ساخت یک شیء از روی کلاس

FuelConsumption کرده و نام این شیء را objectOne به معنی "شیء شماره یک" می گذاریم. در این مرحله کد ما می بایست به صورت زیر باشد:

```
FuelConsumption.java  *ActionClass.java
1 public class ActionClass {
2
3     public static void main(String[] args) {
4         FuelConsumption objectOne = new FuelConsumption(name, carName);
5
6     }
7 }
```

در واقع برای ساخت شیء objectOne که در تصویر فوق مشاهده می شود، پس از نوشتن کلید واژه new و نوشتن حرف F که حرف اول کلاس FuelConsumption می باشد، اکلیپس به شکل زیر پنجره پیشنهادات را برای ما باز می کند:



که با کلیک کردن روی اولین گزینه، اکلیپس به طور خودکار نام پارامترهایی که برای این کلاس در نظر گرفته شده اند را نیز داخل پرانتزها وارد می کند اما این در حالی است که همانطور که در تصویر قبل ملاحظه می شود اکلیپس دور نام پارامترها نقطه چین قرار داده و در کنار نام فایل ActionClass.java نیز یک علامت ستاره قرمز رنگ دیده می شود. به طور خلاصه این نقطه چین ها و ستاره قرمز رنگ حاکی از آن هستند که مقدار اختصاص داده شده به پارامترهای داخل

دوره آموزش جاوا

کلیه حقوق متعلق به وب سایت نردبان است.

مدرس: بهزاد مرادی

پرانتر به درستی وارد نشده اند. در حقیقت در متد Constructor ما دو پارامتر از جنس کلاس String در نظر گرفته و همانطور که قبلا توضیح داده شده است، Value های اختصاص داده شده با String ها می بایست داخل دو علامت “ “ قرار گیرند. حال با دانستن این نکته، به محض اینکه پارامترهای شیء ساخته شده را داخل علامت “ “ قرار دهیم و برنامه خود را Save کنیم خواهیم دید که ستاره قرمز رنگ و نقطه چین ها از بین خواهند رفت:

```

1 public class ActionClass {
2
3     public static void main(String[] args) {
4         FuelConsumption objectOne = new FuelConsumption('name', 'carName');
5
6     }
7 }
8

```

از آنجا که این دو مقدار اختصاص یافته به عنوان پارامتر دارای نامی گویا نیستند، اسامی آن ها را تغییر داده و نامی دلخواه برای آن ها در نظر می گیریم (به طور مثال به جای name از واژه Behzad و به جای carName از واژه 206 استفاده می کنیم). اکنون کد ما می بایست به صورت زیر باشد:

```

public class ActionClass {

    public static void main(String[] args) {
        FuelConsumption objectOne = new FuelConsumption("Behzad", "206");
    }
}

```

در حقیقت با این کار مقدار Behzad و 206 برای پارامترهای به ترتیب name و carName موجود در Constructor در نظر گرفته می شوند و از آنجا که پارامترهای موجود در Constructor برابر با Field های به ترتیب name و carName قرار داده شده اند، پس مقادیر Behzad و 206 به Field های مرتبط ارسال می شوند. حال ممکن است که این سوال برای مخاطبین پیش آید که چه طور می توان مقادیر این پارامترها را به صورت دینامیک وارد برنامه کرد که پاسخ به این سوال بسیار ساده است: به کار گیری کلاس Scanner. برای این منظور کلاس ActionClass را به شکل زیر تکمیل می کنیم:

```
import java.util.Scanner;

public class ActionClass {

    public static void main(String[] args) {
        Scanner keyboardInput = new Scanner(System.in);
        String userName = keyboardInput.next();
        String userCarName = keyboardInput.next();
        FuelConsumption objectOne = new FuelConsumption(userName,
userCarName);
    }
}
```

همانطور که در کد فوق ملاحظه می شود یک شیء از روی کلاس Scanner تحت عنوان keyboardInput به معنی **"ورودی کیبورد"** ساخته ایم (برای آشنایی بیشتر با کلاس Scanner به آموزش های یازدهم، دوازدهم و سیزدهم مراجعه نمایید). سپس دو Instance یا نمونه از روی کلاس String تحت عناوین userName و userCarName به معانی به ترتیب **"نام کاربر"** و **"نام خودروی کاربر"** تعریف کرده ایم. حال به جای آنکه مقادر پارامترهای شیء ساخته شده از روی کلاس FuelConsumption را به صورت دستی وارد کنیم، صرفا نیاز است تا نام این دو شیء ساخته شده از روی کلاس String را جای پارامترها قرار دهیم و همانطور که مشاهده می شود داخل پرانتز به جای عبارت Behzad نام userName و به جای عبارت 206 نام userCarName قرار گرفته است. اکنون ما نیاز داریم تا با کاربر خود تعامل داشته باشیم و یکسری اطلاعات به وی داده تا بر آن اساس کاربر داده های خود را وارد برنامه کند. برای این منظور کد خود را به شکل زیر تکمیل می کنیم:

```
import java.util.Scanner;

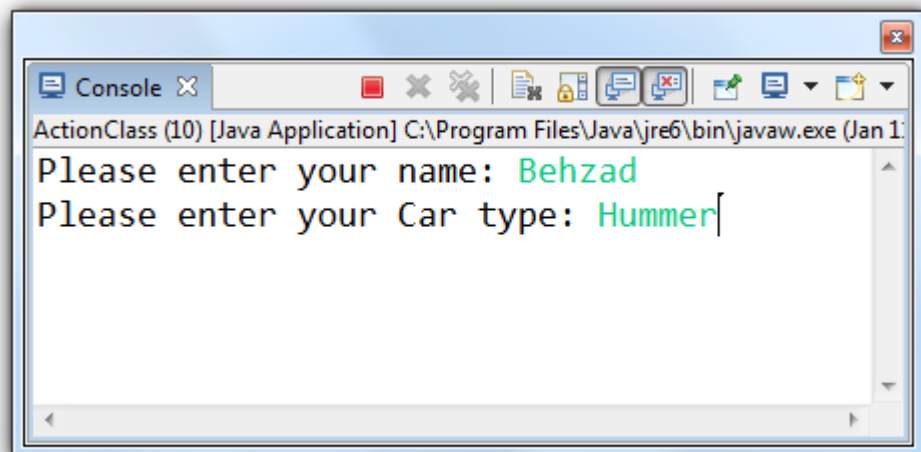
public class ActionClass {

    public static void main(String[] args) {
        Scanner keyboardInput = new Scanner(System.in);
        System.out.print("Please enter your name: ");
        String userName = keyboardInput.next();
        System.out.print("Please enter your Car type: ");
        String userCarName = keyboardInput.next();
        FuelConsumption objectOne = new FuelConsumption(userName,
userCarName);
    }
}
```

به طور خلاصه با قرار دادن دستور `System.out.print("Please enter your name: ");` روی صفحه نمایش عبارت `Please enter your name:` به معنی **"لطفا نام خود را وارد نمایید"** نمایش داده خواهد شد. سپس دستور `String userName = keyboardInput.next();` را قرار داده تا پس از به نمایش آمدن عبارت فوق هر آنچه کاربر به عنوان نام خود تایپ کرد مقابل عبارت `Please enter your name:` قرار گیرد (در حقیقت از آنجا که از متد `print` استفاده کرده ایم نشانگر موس برخلاف دستور `println` به خط بعد انتقال نخواهد یافت).

به همین منوال عبارت `Please enter your car type` را نیز روی صفحه نمایش نشان خواهیم داد و دستور `String userCarName = keyboardInput.next();` را پس از آن قرار می دهیم. اکنون یک بار برنامه را اجرا می کنیم:





همانطور که ملاحظه می شود در اجرای فوق بنده اول نام خود را وارد کرده سپس نام یک خودرو که در اینجا هامر است را مد نظر قرار داده ام (به محض انجام این کار، این دو نام به عنوان پارامترهای Constructor و بالتبع مقادیر Field های کلاس قبلی مد نظر قرار داده خواهند شد). پس از آنکه ما نام خودروی خود را وارد کنیم و دکمه Enter را فشار دهیم، برنامه دستور دیگری را اجرا نخواهد کرد چرا که دستور دیگری برای برنامه تعریف نکرده ایم. اکنون که توانستیم با موفقیت مقادیر پارامترهای ورودی برنامه را مشخص کنیم، حال نوبت به گرفتن اطلاعات تکمیلی از کاربر پیرامون مسافت طی کرده با خودرو و همچنین مقدار بنزین موجود در باک خودروی وی می رسد. برای این کار کد برنامه را به شکل زیر تکمیل می کنیم:



```

import java.util.Scanner;

public class ActionClass {

    public static void main(String[] args) {
        Scanner keyboardInput = new Scanner(System.in);
        System.out.print("Please enter your name: ");
        String userName = keyboardInput.next();

        System.out.print("Please enter your Car type: ");
        String userCarName = keyboardInput.next();

        FuelConsumption objectOne = new FuelConsumption(userName,
userCarName);

        // Code related to distance and fuel amount
        System.out.print("How many kilometers did you go \nwith your
car: ");
        double distance = keyboardInput.nextDouble();

        System.out.print("How much petrol did your car have? ");
        double petrolUsed = keyboardInput.nextDouble();
    }
}

```

در واقع یک Comment حاوی عبارت Code related to distance and fuel amount به معنی "کد مرتبط با مسافت و مقدار سوخت" قرار می دهیم تا در حین بازبینی کد خود دچار سردرگمی نشویم سپس قصد داریم تا عبارت How many kilometers did you go with your car? به معنی "شما چند کیلومتر با خودروی خود رفتید؟" را روی صفحه مانیتور نمایش دهیم. سپس یک متغیر از جنس double تحت عنوان distance به معنی "مسافت" ایجاد کرده و قصد داریم تا مقدار آن را به صورت دینامیک وارد برنامه کنیم از این رو پس از علامت مساوی نام شیئی ساخته شده از روی کلاس Scanner را نوشته و سپس متد nextDouble را فرا می خوانیم که وظیفه آن گرفتن مقادیر اعشاری از طریق کیبورد است (لازم به ذکر است که اگر کاربر یک عدد صحیح وارد برنامه کند مثلاً 90 کیلومتر، برنامه به صورت خودکار آن را به یک عدد اعشاری تبدیل کرده و عدد 90.0 را وارد برنامه خواهد کرد).

به همین منوال عبارت `How much petrol did your car have?` به معنی "خودروی شما چقدر بنزین داشت؟" را روی صفحه مانیتور نمایش داده و سپس متغیری دیگری از جنس `double` تحت عنوان `petrolUsed` به معنی "**مقدار بنزین مصرف شده**" تعریف کرده و مقدار اولیه آن را مساوی با نام شیئی ساخته شده از روی کلاس `Scanner` قرار داده و سپس متد `nextDouble` را فرا می خوانیم که وظیفه آن گرفتن مقادیر اعشاری از طریق کیبورد است. اکنون یک بار دیگر برنامه خود را اجرا می کنیم:

```
<terminated> ActionClass (10) [Java Application] C:\Program Files\Java\jre6\bin\ja
Please enter your name: Behzad
Please enter your Car type: 206
How many kilometers did you go
with your car 145
How much petrol did your car have? 45
|
```

همانطور که ملاحظه می شود برنامه به درستی کار کرده و اطلاعات به درستی وارد برنامه

شدند(همانطور که در تصویر فوق ملاحظه می شود سوال سوم که `How many kilometers did you go with your car?` است از واژه `with` به بعد به خط بعد انتقال پیدا کرده است که این کار توسط دستور \ به علاوه حرف `n` صورت گرفته است که هدف از اینکار صرفا خوانایی بیشتر و بهتر برنامه بوده است).

اکنون زمان آن فرا رسیده است تا فرمولی برای به دست آوردن مصرف سوخت بنویسیم:

$$\frac{\Delta}{x} = \frac{115}{100}$$

همانطور که در تصویر فوق ملاحظه می شود بنده فرض کرده ام که با هشت لیتر بنزین می توانیم مسافت ۱۱۵ کیلومتر را طی کنم. حال می خواهیم ببینم که مسافت ۱۰۰ کیلومتر با چه مقدار بنزین امکان پذیر است. برای این کار عدد ۱۰۰ را در عدد ۸ ضرب کرده و بر عدد ۱۱۵ تقسیم می کنیم که با اینکار مقدار مجهول ایکس را به دست خواهیم آورد.

پس برای اینکه بتوانیم مصرف سوخت را برای برنامه خود به دست آوریم نیاز است تا عدد صد را در مقدار متغیر petrolUsed ضرب کرده و حاصل ضرب این دو عدد را بر مقدار متغیر distance تقسیم کنیم. برای اینکار کد خود را به شکل زیر تکمیل می کنیم:

```

import java.util.Scanner;

public class ActionClass {

    public static void main(String[] args) {
        Scanner keyboardInput = new Scanner(System.in);
        System.out.print("Please enter your name: ");
        String userName = keyboardInput.next();

        System.out.print("Please enter your Car type: ");
        String userCarName = keyboardInput.next();

        FuelConsumption objectOne = new FuelConsumption(userName,
userCarName);

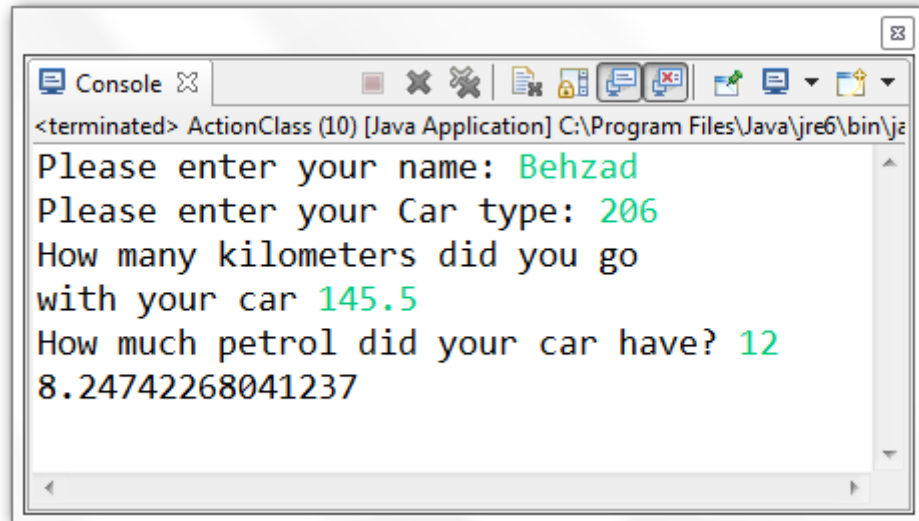
        // Code related to distance and fuel amount
        System.out.print("How many kilometers did you go \nwith your
car: ");
        double distance = keyboardInput.nextDouble();

        System.out.print("How much petrol did your car have? ");
        double petrolUsed = keyboardInput.nextDouble();

        double result = 100 * petrolUsed / distance;
        System.out.println(result);
    }
}

```

همانطور که ملاحظه می شود در دو خط آخر کد، متغیر result به معنی "نتیجه" تعریف شده و فرمول محاسبه مصرف در هر صد کیلومتر برای آن در نظر گرفته شده است سپس مقدار این متغیر در دستور System.out.println قرار گرفته است. حال مجدد برنامه خود را اجرا می کنیم:



```
<terminated> ActionClass (10) [Java Application] C:\Program Files\Java\jre6\bin\j...
Please enter your name: Behzad
Please enter your Car type: 206
How many kilometers did you go
with your car 145.5
How much petrol did your car have? 12
8.24742268041237
```

همانطور که ملاحظه می شود پس از طی مسافتی معادل با 145.5 کیلومتر و مصرف ۱۲ لیتر بنزین برنامه مقدار 8.24 را در هر صد کیلومتر محاسبه کرده است (در واقع خودروی ۲۰۶ مصرفی معادل با ۷ لیتر در هر صد کیلومتر دارد. شاید علت افزایش مصرف سوخت استفاده از کولر باشد!).

همانطور که در اجرای فوق مشاهده می شود برنامه یک عدد بسیار دقیق از مقدار مصرف سوخت ما در هر صد کیلومتر ارائه می دهد که شاید برای خیلی از کاربران عددی به این دقت خیلی مهم نباشد و صرفا بخواهند بدانند که مثلا مصرف خودروی ایشان معادل با ۸ لیتر در هر صد کیلومتر بوده است. برای این کار می توان از دستور `Math.round` استفاده کرد به این صورت که مقدار اختصاص داده شده به این دستور رند خواهد شد. برای این منظور کد خود را به شکل زیر تکمیل می کنیم:

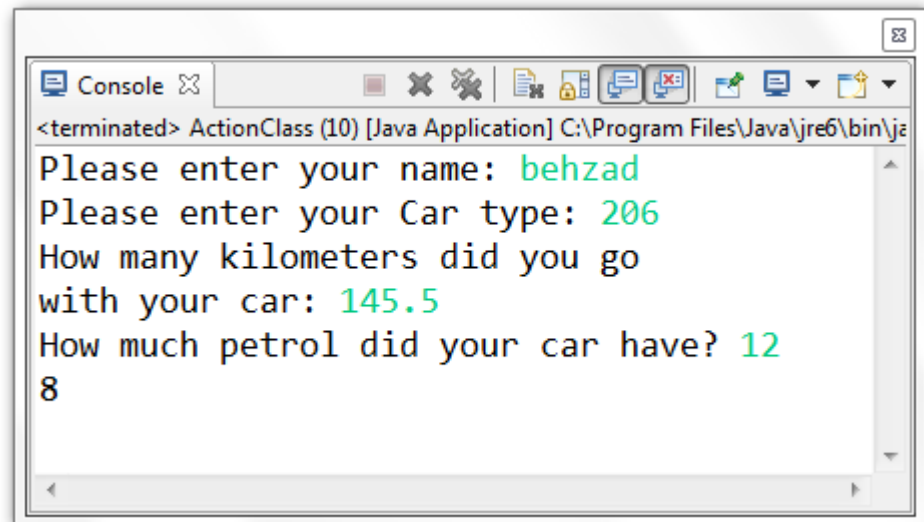
```
double result = 100 * petrolUsed / distance;
System.out.println(Math.round(result));
```

به طور خلاصه با قرار دادن کلاس `Math` و ضمیمه کردن متد `round` به آن و قرار دادن متغیری که می خواهیم مقدار آن رند شود (منظور متغیر `result` است) در داخل پرانتز مقابل متد `round` می توانیم به نتیجه دلخواه برسیم. اکنون برنامه را با همان اطلاعات ورودی در اجرای فوق مجدد اجرا می کنیم:

دوره آموزش جاوا

کلیه حقوق متعلق به وب سایت نردبان است.

مدرس: بهزاد مرادی



```
<terminated> ActionClass (10) [Java Application] C:\Program Files\Java\jre6\bin\ja
Please enter your name: behzad
Please enter your Car type: 206
How many kilometers did you go
with your car: 145.5
How much petrol did your car have? 12
8
```

می بینیم که خروجی برنامه کاملاً رند شده است و برنامه به ما می گوید که پس از طی 145.5 کیلومتر مسافت و مصرف ۱۲ لیتر بنزین، ما در هر صد کیلومتر فقط هشت لیتر بنزین مصرف کرده ایم. اکنون برای تکمیل برنامه ما می توانیم به جای فقط نمایش دادن مقدار متغیر result، به برنامه خود دستور دهیم که مثلاً عبارت Dear Behzad, your 206 car used 8 liters petrol per 100 kilometers. به معنی "بهزاد جان، خودروی ۲۰۶ شما در هر صد کیلومتر ۸ لیتر بنزین مصرف می کند." را روی صفحه نمایش نشان دهد. برای این کار به سادگی می توانیم نام متغیرهای مرتبط با نام، نام خودرو و نتیجه را در دستور System.out.println قرار دهیم و به نتیجه دلخواه برسیم اما از آنجا که در کلاس اول خود یکسری متد بازگشتی تعریف کردیم حال قصد داریم تا برای درک بهتر مفهوم return از آن متد ها در خروجی برنامه خود استفاده کنیم. برای این منظور کد خود را به شکل زیر تکمیل می کنیم:

```

import java.util.Scanner;

public class ActionClass {

    public static void main(String[] args) {
        Scanner keyboardInput = new Scanner(System.in);
        System.out.print("Please enter your name: ");
        String userName = keyboardInput.next();

        System.out.print("Please enter your Car type: ");
        String userCarName = keyboardInput.next();

        FuelConsumption objectOne = new FuelConsumption(userName,
userCarName);

        // Code related to distance and fuel amount
        System.out.print("How many kilometers did you go \nwith your car: ");
        double distance = keyboardInput.nextDouble();

        System.out.print("How much petrol did your car have? ");
        double petrolUsed = keyboardInput.nextDouble();

        double result = 100 * petrolUsed / distance;
        System.out.println();

        System.out.println("Dear " + objectOne.returnName() + ", your "
            + objectOne.returnCarName() + " car \nconsumes "
            + Math.round(result) + " liters petrol in 100 kilometers.");
    }
}

```

در واقع تنهای تغییری که در کد فوق صورت گرفته است دستور هایی است که داخل `System.out.println` قرار گرفته است. در ابتدا عبارت `Dear` به معنی "عزیز" را داخل دو علامت " " قرار داده و سپس یک علامت + می گذاریم. سپس شیئی که از روی کلاس `FuelConsumption` تحت عنوان `objectOne` ساخته ایم را نوشته و یک نقطه پس از آن قرار می دهیم. در واقع پس از قرار دادن نقطه به کلیه متدهای کلاسی که این شی از روی آن ساخته شده است دسترسی خواهیم داشت. سپس متدی که قرار بود مقدار `Field` مرتبط با نام را بازگرداند را می نویسیم. به عبارت دیگر `objectOne.returnName()` سپس یک علامت به علاوه قرار داده و عبارت `your` , به معنی "مال شما" را می نویسیم. از آنجا که می خواهیم بگویم مثلاً ماشین پژو شما، پس نیاز به نام ماشین داریم. برای این منظور مجدد نام شیئی ساخته شده از روی کلاس `FuelConsumption` را نوشته و پس از قرار دادن یک نقطه نام متدی را

دوره آموزش جاوا

کلیه حقوق متعلق به وب سایت نردبان است.

مدرس: بهزاد مرادی



می نویسیم که قرار بود مقدار Field مرتبط با نام خودرو را بازگرداند. به عبارت دیگر نام متد `returnCarName()` را می نویسیم. سپس یک علامت به علاوه قرار داده و عبارت `car` consumes به معنی **"ماشین استفاده می کند"** را می نویسیم (لازم به ذکر است از آنجا که نوشته ما طولانی خواهد شد، برای خوانایی بیشتر می توان از علامت \ به علاوه حرف n در هر جایی از دستور استفاده کنیم که می خواهیم از آن نقطه به بعد به خط بعد منتقل شود). سپس نام متغیر `result` که قرار بود نتیجه نهایی را اعلام کند می نویسیم. در این کد برای آنکه عدد متغیر `result` رند شود از کلاس `Math` و متد `round()` استفاده می کنیم و نام متغیر `result` را داخل پرانتز متد `round()` می نویسیم. آخرین جمله ای که در دستور ما قرار می گیرد عبارت `liters petrol in 100 kilometers` به معنی **"... لیتر بنزین در هر صد کیلومتر"** می باشد. اکنون برنامه تکمیل شده و می توانیم برنامه را اجرا کنیم:

```

<terminated> ActionClass (10) [Java Application] C:\Program Files\Java\jre6\bin\jav
Please enter your name: Behzad
Please enter your Car type: 206
How many kilometers did you go
with your car: 180
How much petrol did your car have? 13
|
Dear Behzad, your 206 car
consumes 7 liters petrol in 100 kilometers

```

همانطور که در تصویر فوق ملاحظه می شود برنامه به درستی کار کرده و آخرین دستوری که اضافه کردیم عبارت `Dear Behzad, your 206 car consumes 7 liters petrol in 100 kilometers` به معنی **"بهزاد عزیز، ماشین ۲۰۶ شما در هر ۱۰۰ کیلومتر مقدار ۷ لیتر بنزین مصرف می کند"** را به ما نشان می دهد. جهت یادآوری عرض می کنیم که در این دستور آخر مقادیر `Behzad` و `206` از طریق متدهای بازگشتی کلاس اول فرا خوانده شده اند.

پس از مطالعه این آموزش انتظار می رود بتوانیم به سؤالات زیر پاسخ بدهیم:

۱. چرا از دستور this می بایست استفاده کرد؟
۲. وظیفه کلاس Math چیست؟
۳. از چه متدهایی با کلاس Math می توان استفاده کرد؟
- ۴.

دوره آموزش جاوا

کلیه حقوق متعلق به وب سایت نردبان است.

مدرس: بهزاد مرادی