**Title of script: Object Oriented concept in Ruby**
**Author: Anjana Nair**

**Keywords: class**

| **Visual Cue** | **Narration** |
|---|---|
| Slide 1 | Welcome to this spoken tutorial on **Object Oriented Programming – Methods** in **Ruby**. |
| Slide 2 | In this tutorial we will learn to use:<br>• **instance methods**<br>• **class methods**<br>• **accessor methods** |
| Slide 3 | Here we are using<br>• **Ubuntu** version 12.04<br>• **Ruby**1.9.3 |
| Slide 4<br>Pre-requisites | To follow this tutorial, you must have a working **Internet** Connection.<br><br>You must also have knowledge of **Linux** commands, **Terminal** and **Text-editor.**<br><br>If not, for relevant tutorials, please visit our website. |
| | Before we begin, recall that we had created "**ttt**" directory earlier.<br><br>Let's go to that directory. |
| Switch to the **terminal** which has all the commands for creating the directories and the prompt should be in **oop-methods** directory | Then to **ruby-tutorial.**<br>**Create a directory names oop-methods and cd into it**. |
| Slide 5<br>What are **instance methods**? | What are **Instance methods**?<br><br>**Instance methods** are those **methods** that are available to all **instances** of the **class.**<br><br>Earlier we had studied how to create **objects** or **instances** of a **class**. |
| Switch to **gedit** where you have already opened the file **"instance_methods.rb"** with the class defnition code typed inside. | Create a new file in **gedit** as shown in the basic level **Ruby** tutorials.<br><br>Name it **instance_methods.rb** |
| | I have a working example of the implementing **instance methods**. |

| | |
|---|---|
| | You can pause the tutorial, and type the code as we go through it. |
| Highlight the "**initialize**" method block | I have defined a **class** named **Product** in this example.<br><br>I have called an **initialize method** to initialize the **instance variables** "**name**" and "**price**". |
| Highlight the methods "**name**" and "**price**" | I have also defined **instance methods** named "**name**" and "**price**".<br><br>Each of them return **instance variables "@name**" and **"@price**" respectively. |
| | **Instance methods** are defined just like normal **methods**.<br><br>Earlier we had studied how to create **methods** in **Ruby**.<br><br>Shortly we will see how these **methods** will be available to all **instances.** |
| | Now let us implement the logic we have. |
| Highlight the poduct object creation code. | Here, I have initialized a **Product object** and named it as **product_object_1**.<br><br>I have initialized it with a name value and a price value. |
| Highlight the **initialize** block. | The **initializer block** passes the values to the **instance variables @name** and **@price**.<br><br>~~Earlier we had studied about **instance variables**.~~ |
| Highlight **:**<br>**puts product_object_1.name**<br>**puts product_object_1.price** | Now this **product instance** or **object** can use the **instance methods name** and **price**.<br><br>On invoking these **methods,** we get the values stored in the **instance variables**. |
| On the **terminal** type-<br>**ruby instance_methods.rb** >> press **Enter** | Now let us execute this code.<br><br>Switch to the **terminal** and type:<br>**ruby instance_methods.rb**<br><br>and press **Enter** to see the output. |
| Point to the output. | You will see that it will print the values you initialized the **object** with. |

| | |
|---|---|
| | Namely, **laptop** and **35,000**. |
| | Next let us switch back to **gedit**. |
| Highlight the initialization of **product_object_2** | Now let us initialize another **instance** or **object**.<br><br>Let us name this object **product_object_2**.<br><br>This time let us give a different set of values for name and price. |
| Highlight the **puts** statements | Now let us call the **instance methods** "**name**" and "**price**" for this **object.** |
| Switch to the **terminal** >> Press up-arrow >> **Enter.** | Next let us switch back to the terminal and execute the code like before. |
| Output on the **terminal.** | You will notice that it executes successfully and prints out the new values.<br><br>This proves that **instance methods** are available to all **objects** of the **class Product**. |
| | You should now be able to write your own **instance methods.** |
| | Next let us look at what **class methods** are. |
| Slide 6<br>What are **class methods** | **Class methods** are **methods** available only to the **class.**<br><br>These **methods** are not available to instances of the **class.**<br>There are different ways you can define class methods. |
| | Let us look at an example. |
| Switch to **gedit** where you have already opened the file **"class_methods.rb"** with the class defnition code typed inside. | Create a new file in **gedit** as shown in the basic level **Ruby** tutorials.<br><br>Name it **class_methods.rb** |
| | I have a working example of **class methods**.<br><br>You can pause the tutorial, and type the code as we go through it. |
| Highlight **Product.** | I have defined a **Product class** like before. |
| Highlight the **initialize** code block. | I have also called an initializer like before.<br><br>However, this time I have added an extra **argument** called **description**. |

| | |
|---|---|
| | I am also using class variables to hold the values unlike instance variables earlier. |
| Highlight the first class declaration code. | This **class** will demonstrate to you the 3 different ways one can define **class methods.**<br><br>Checkout the **class method** declaration for **name**.<br><br>Here it is defined using the **class name Product**. |
| Highlight the second class declaration code. | Then checkout the second **class methods** declaration.<br><br>Here I have used the **self keyword**. |
| Highlight the second class declaration code. | Next, checkout the third way you can define **class methods**. |
| | Now let us implement these **class methods**. |
| Highlight the object creation logic. | Let us first initialize an object of **Product** like before.<br><br>This time we are also giving a value for the **description**. |
| Highlight the lines corresponding the class method invocation. | Now let us invoke the **class methods** as shown here. |
| | Now let us execute the code and inspect the output. |
| Switch to the **terminal** >> type **ruby class_methods.rb** >> **Enter.** | Switch to the terminal and execute the code like before. |
| Output on the **terminal.** | You will notice that it will print the values for **name, price** and **description.** |
| | Now you should be able to write your own class methods.<br><br><<PAUSE>> |
| | Next we shall see what **accessor methods** are. |
| Slide 5<br>What are **accessor methods** | **Ruby** uses **accessor methods** to access data defined within **classes**.<br><br>Accessor methods comprise of **setter methods** and **getter methods**.<br><br>**Setter methods** set the values. |

| | |
|---|---|
| | **Getter methods** get those values.<br><br>**Ruby** uses the **keyword attr_accessor** to declare these **methods**. |
| | Let us look at an example of **accessor methods**. |
| Switch to **gedit** where you have already opened the file **"accessor_methods.rb"** with the class defnition code typed inside. | Create a new file in **gedit** as shown in the basic level **Ruby** tutorials.<br><br>Name it **accessor_methods.rb** |
| | I have a working example of the implementing **accessor methods**.<br><br>You can pause the tutorial, and type the code as we go through it. |
| Highlight "**attr_accessor**" line | I have defined a **class** named **Product** in this example.<br><br>I have declared **attr_accessor** for **name and price.**<br><br>That is all that is required to use these types of **methods**. |
| Highlight object creation logic. | Now let us put it to action.<br><br>I have initialized a **Product object**. |
| Highlight the **setter** logic. | Then I have set the **name** and **price** of the **product object**.<br><br>This is possible because the **attr_declaration,** by default, creates **methods** for setting values. |
| Highlight the **getter** logic. | I have then attempted to print the values using the **getter methods** for **name** and **price**.<br><br>These **getter methods** were also generated by the declaration of **attr_accessor**. |
| | Now let us execute the code like before. |
| | You will notice that it prints the values that were set. |
| | By now you should be able to write your own **accessor methods**. |
| **Slide**<br>**Accessor Methods** | One thing to note is that **accessor methods** are, by default, **instance methods.**<br><br>Thus they can be accessed by different **instances** |

| | |
|---|---|
| | of the **class Product**. |
| Slide<br>Summary | In this tutorial we have learnt about:<br>• **instance methods**<br>• **class methods** and<br>• **accessor methods** |
| Slide<br>Assignment | As an assignment:<br>• Define a **class** named **Temperature**<br>• Write an **instance method** using **Ruby's accessor method** syntax.<br>• This **method** should calculate the **Celsius** for the given **Fahrenheit.** |
| Slide<br>About the Spoken Tutorial Project | Watch the video available at the following link.<br>It summarizes the Spoken Tutorial project.<br>If you do not have good bandwidth, you can download and watch it. |
| Slide | The Spoken Tutorial Project Team :<br><br>Conducts workshops using spoken tutorials<br>Gives certificates to those who pass an online test<br>For more details, please write to<br>contact at spoken hyphen tutorial dot org |
| Slide<br>Acknowledgments | Spoken Tutorial Project is a part of the Talk to a Teacher project.<br><br>It is supported by the National Mission on Education through ICT, MHRD, Government of India.<br><br>More information on this Mission is available at: spoken hyphen tutorial dot org slash NMEICT hyphen Intro. |
| Previous Slide | This is Anjana Nair signing off. Thank you |