Partially applied functions

Partially applied functions ed functions

Say you have a method with 3 arguments

```
scala> def sum(a: Int, b: Int, c: Int) = a + b + c
sum: (a: Int, b: Int, c: Int)Int

scala> sum(1, 2, 3)
res8: Int = 6
```

```
scala>  val b = sum(1, _: Int, 3)
b: Int => Int = <function1>
```

You can create a function variable with some arguments applied

```
scala>  val b = sum(1, _: Int, 3)
b: Int => Int = <function1>
```

```
scala> b(3)
res10: Int = 7
```

Since only one argument is missing, the Scala compiler generates a new function object that takes one argument

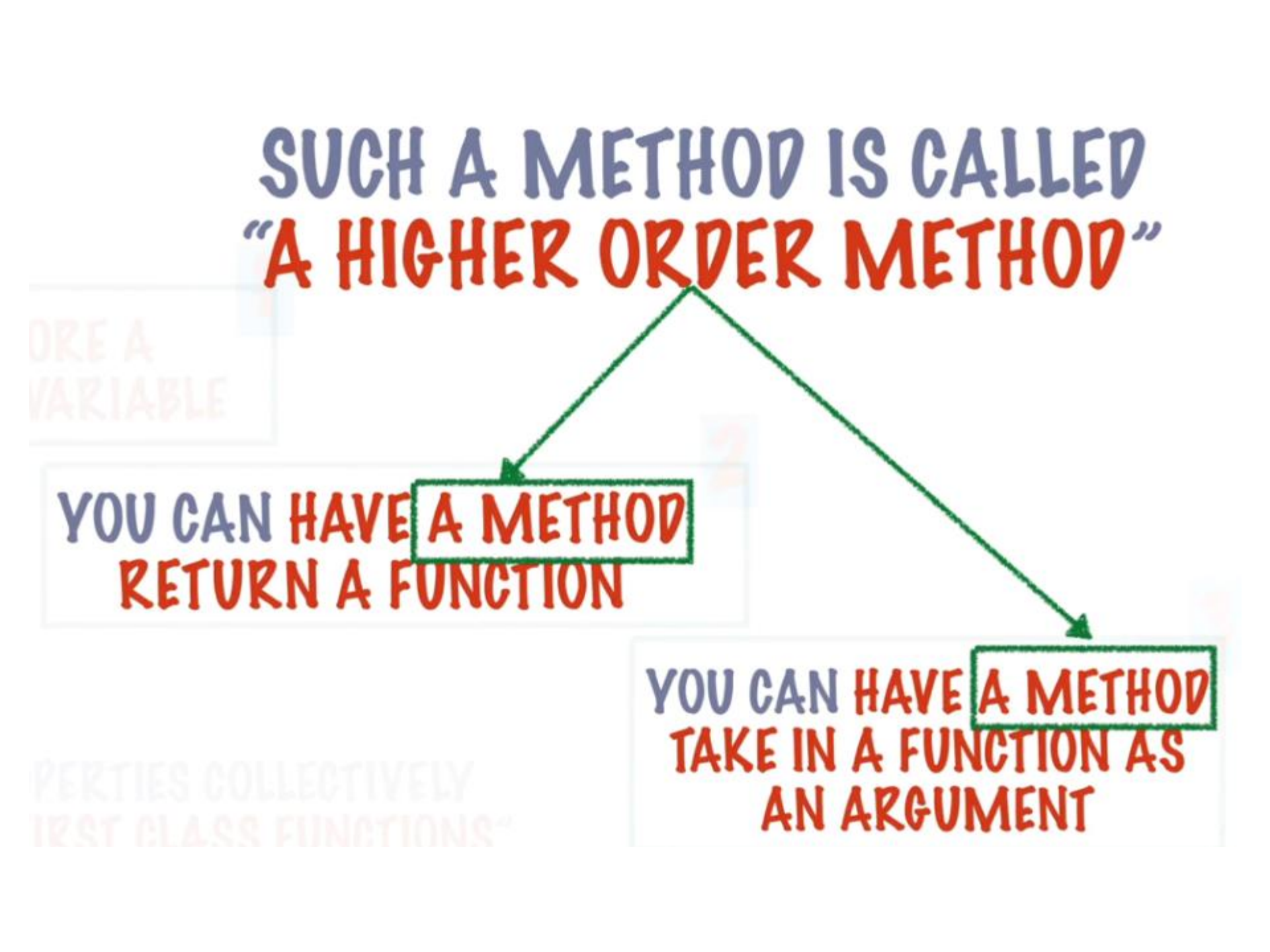
FUNCTION OBJECTS IN SCALA CAN BE TREATED JUST LIKE NUMBERS OR STRINGS

YOU CAN STORE A FUNCTION IN A VARIABLE

YOU CAN HAVE A METHOD RETURN A FUNCTION

YOU CAN HAVE A METHOD TAKE IN A FUNCTION AS AN ARGUMENT

THESE 3 PROPERTIES COLLECTIVELY ARE CALLED "FIRST CLASS FUNCTIONS"


SUCH A METHOD IS CALLED
"A HIGHER ORDER METHOD"
YOU CAN HAVE A METHOD RETURN A FUNCTION
YOU CAN HAVE A METHOD TAKE IN A FUNCTION AS AN ARGUMENT

YOU CAN HAVE A METHOD RETURN A FUNCTION

SAY YOU WANTED TO PRINT A GREETING TO A USER

"HELLO SWETHA"

"NAMASTE JANANI"

"BONJOUR VITTHAL"

YOU CAN HAVE A METHOD
RETURN A FUNCTION

YOU WANT THE GREETING TO CHANGE
BASED ON THE LANGUAGE OF THE USER

"HELLO SWETHA"	ENGLISH
"NAMASTE JANANI"	HINDI
"BONJOUR VITTHAL"	FRENCH

YOU CAN HAVE A METHOD RETURN A FUNCTION

HAVE A METHOD RETURN A FUNCTION BASED ON THE LANGUAGE!

```
def greeting(lang: String)= {
  lang match {
    case "English" => (x: String) => println("Hello "+x)
    case "Hindi" => (x: String) => println("Namaste "+x)
    case "French" => (x: String) => println("Bonjour "+x)
    case "Spanish" => (x: String) => println("Hola "+x)
  }
```

YOU CAN HAVE A METHOD RETURN A FUNCTION

MATCH IS SIMILAR TO SWITCH IN JAVA

```
def greeting(lang: String)= {
  lang match {
     case "English" => (x: String) => println("Hello "+x)
     case "Hindi" => (x: String) => println("Namaste "+x)
     case "French" => (x: String) => println("Bonjour "+x)
     case "Spanish" => (x: String) => println("Hola "+x)
  }
```

YOU CAN HAVE A METHOD RETURN A FUNCTION

THIS IS A FUNCTION OBJECT

```
def greeting(lang: String)= {
  lang match {
    case "English" => (x: String) => println("Hello "+x)
    case "Hindi" => (x: String) => println("Namaste "+x)
    case "French" => (x: String) => println("Bonjour "+x)
    case "Spanish" => (x: String) => println("Hola "+x)
  }
```

YOU CAN HAVE A METHOD RETURN A FUNCTION

FOR EACH LANGUAGE, RETURN A FUNCTION THAT PRINTS THE APPROPRIATE GREETING

```
def greeting(lang: String)= {

  lang match {

    case "English" => (x: String) => println("Hello "+x)

    case "Hindi" => (x: String) => println("Namaste "+x)

    case "French" => (x: String) => println("Bonjour "+x)

    case "Spanish" => (x: String) => println("Hola "+x)

  }
```

YOU CAN HAVE A METHOD RETURN A FUNCTION

```
def greeting(lang: String)= {
  lang match {
    case "English" => (x: String) => println("Hello "+x)
    case "Hindi" => (x: String) => println("Namaste "+x)
    case "French" => (x: String) => println("Bonjour "+x)
    case "Spanish" => (x: String) => println("Hola "+x)
```

```
def main (args: Array[String]){

  val greetEnglish = greeting("English")
  greetEnglish("Swetha")

  val greetSpanish = greeting("Spanish")
  greetSpanish("Janani")

}
```

YOU CAN GET THE APPROPRIATE FUNCTION BASED ON THE LANGUAGE OF THE USER

YOU CAN HAVE A METHOD RETURN A FUNCTION

```
def greeting(lang: String)= {
  lang match {
    case "English" => (x: String) => println("Hello "+x)
    case "Hindi" => (x: String) => println("Namaste "+x)
    case "French" => (x: String) => println("Bonjour "+x)
    case "Spanish" => (x: String) => println("Hola "+x)

def main (args: Array[String]){
    val greetEnglish = greeting("English")
    greetEnglish("Swetha")

    val greetSpanish = greeting("Spanish")
    greetSpanish("Janani")

}
```

GREETENGLISH IS A FUNCTION OBJECT FOR GREETING ENGLISH SPEAKING USERS

YOU CAN HAVE A METHOD RETURN A FUNCTION

```
def greeting(lang: String)= {
  lang match {
    case "English" => (x: String) => println("Hello "+x)
    case "Hindi" => (x: String) => println("Namaste "+x)
    case "French" => (x: String) => println("Bonjour "+x)
    case "Spanish" => (x: String) => println("Hola "+x)
 def main (args: Array[String]){
    val greetEnglish = greeting("English")
    greetEnglish("Swetha")

    val greetSpanish = greeting("Spanish")
    greetSpanish("Janani")

 }
```

CALL THE GREETING METHOD AND STORE THE VALUE RETURNED IN GREETENGLISH

YOU CAN HAVE A METHOD RETURN A FUNCTION

```
def greeting(lang: String)= {
  lang match {
    case "English" => (x: String) => println("Hello "+x)
    case "Hindi" => (x: String) => println("Namaste "+x)
    case "French" => (x: String) => println("Bonjour "+x)
    case "Spanish" => (x: String) => println("Hola "+x)

def main (args: Array[String]){

  val greetEnglish = greeting("English")
  greetEnglish("Swetha")

  val greetSpanish = greeting("Spanish")
  greetSpanish("Janani")

}
```

USE IT TO PRINT THE APPROPRIATE GREETING FOR ALL ENGLISH SPEAKING USERS

YOU CAN HAVE A METHOD RETURN A FUNCTION

```
def greeting(lang: String)= {
  lang match {
    case "English" => (x: String) => println("Hello "+x)
    case "Hindi" => (x: String) => println("Namaste "+x)
    case "French" => (x: String) => println("Bonjour "+x)
    case "Spanish" => (x: String) => println("Hola "+x)
  }

def main (args: Array[String]){

  val greetEnglish = greeting("English")
  greetEnglish("Swetha")

  val greetSpanish = greeting("Spanish")
  greetSpanish("Janani")

}
```

IF THE USER IS SPANISH - GET A DIFFERENT FUNCTION AND USE IT