ÖZYEĞİN UNIVERSITY

CS105
Introduction to Object-Oriented Programming

Prof. Dr. Nizamettin AYDIN

naydin@itu.edu.tr

nizamettin.aydin@ozyegin.edu.tr

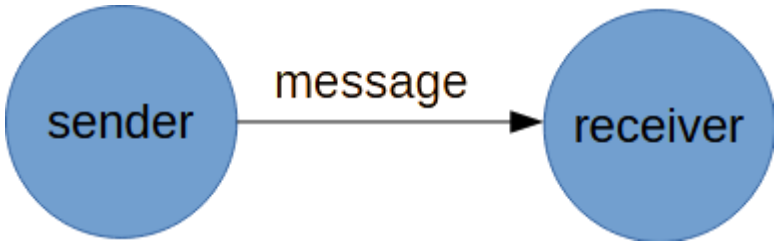
this

Outline

- *this* keyword
- *this* (first use case)
- *this* (second use case)
- Using *this* for constructor call

this

- An important keyword used inside methods in order to refer to the current object.
- ***this*** is "**the calling object**"
  - *this* is the object which called the method
  - The object that occurs before the dot



- In sender:
  - receiver.method(arguments)
- If sender and receiver are the same:
  - method(arguments) [implicit version]
  - this.method(arguments) [explicit version]

Bank Account…

```
Public class Account {
    private int number;
    private double balance;
    private String currency;
    private double interestRate;
    public Account(int n, double b, String c){
        number = n;
        if (b > 0)
            balance = b;
        else
            balance = 0;

        interestRate = 0;
        checksetCurrency(c);
    }
    public Account(int n, String c){
        number = n;
        balance = 0;
        interestRate = 0;

        checksetCurrency(c);
    }
    public Account(int n){
        number = n;
        balance = 0;
        currency = "TL";
        interestRate = 0;
    }
```

…Bank Account

```
    public int getNumber() {
        return number;
    }
    public double getBalance() {
        return balance;
    }
    public String getCurrency() {
        return currency;
    }
    public boolean getInterestRate() {
        return interestRate;
    }
    public void setInterestRate(double i) {
        interestRate = i;
    }
    public void setCurrency(String c) {
        if (currency.equals("TL") && c.equals("USD")) {
            balance = balance / 32.88;
    }
        if (currency.equals("USD") && c.equals ("TL")){
            balance = balance * 32.88;
    }
        if (currency.equals("TL") || c.equals("USD")) {
            currency = c;
    }
}
```

this

- There are several use cases of **this**
  1. It is used to access the instance variables without any confusion with parameter names.

```
public void setInterestRate(double i) {
   interestRate = i;
}
public void setCurrency(String c) {
   if (currency.equals("TL") && c.equals("USD")) {
      balance = balance / 32.88;
}
   if (currency.equals("USD") && c.equals ("TL")){
      balance = balance * 32.88;
}
   if (currency.equals("TL") || c.equals("USD")) {
      currency = c;
}
```

this (first use case)

```
public void setInterestRate(double i) {
   interestRate = i;
}
public void setCurrency(String c) {
   if (currency.equals("TL") && c.equals("USD")) {
      balance = balance / 32.88;
}
   if (currency.equals("USD") && c.equals ("TL")){
      balance = balance * 32.88;
}
   if (currency.equals("TL") || c.equals("USD")) {
      currency = c;
}
```

- One letter parameters are not very descriptive.
  - It is usually the convention in Java to use the instance variable name as parameters in the corresponding set methods.

this (first use case)

- It should be smt like:

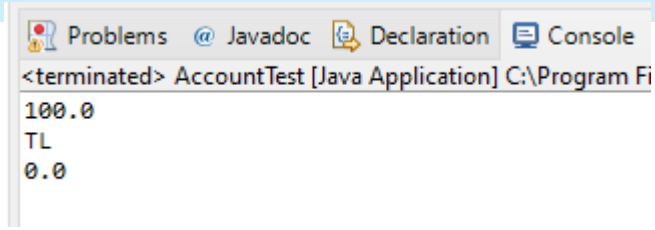
```
public void setInterestRate(double interestRate) {
   interestRate = interestRate;
}
public void setCurrency(String currency) {
   if (currency.equals("TL") && currency.equals("USD")) {
   balance = balance / 32.88;
}
   if (currency.equals("USD") && currency.equals ("TL")){
   balance = balance * 32.88;
}
   if (currency.equals("TL") || currency.equals("USD")) {
   currency = currency;
}
```

- parameter names are same with class instance names
- no compile error

this (first use case)

```
public void setInterestRate(double interestRate) {
    interestRate = interestRate;
}
public void setCurrency(String currency) {
    if (currency.equals("TL") && currency.equals("USD")) {
    balance = balance / 32.88;
}
    if (currency.equals("USD") && currency.equals ("TL")){
    balance = balance * 32.88;
}
    if (currency.equals("TL") || currency.equals("USD")) {
    currency = currency;
}
```

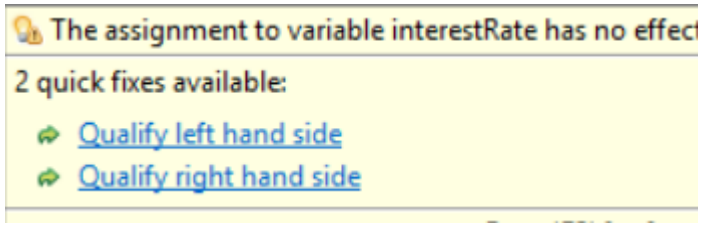
```
public static void main(String[] args) {

    Account account1 = new Account(1, 100, "TL");

    account1.setCurrency("USD");
    System.out.println(account1.getBalance());
    System.out.println(account1.getCurrency());

    account1.setInterestRate(0.02);
    System.out.println(account1.getInterestRate());
}
```



- What should be the output?

this (first use case)

```
public void setInterestRate(double interestRate) {
   interestRate = interestRate;
}
public void setCurrency(String currency) {
   if (currency.equals("TL") && currency.equals("USD")) {
   balance = balance / 32.88;
}
   if (currency.equals("USD") && currency.equals ("TL")){
   balance = balance * 32.88;
}
   if (currency.equals("TL") || currency.equals("USD")) {
   currency = currency;
}
```



- When class instance and parameter have the same names, the assignment operation has no effect

this (first use case)

- The parameters interestRate and currency shadow the class instances.

```
public void setInterestRate(double interestRate) {
   interestRate = interestRate;
```

- The interestRate assigns the parameter interestRate to itself, causing no change whatsoever.

- In such cases you can use the ***this*** keyword to reach the class instances.

```
public void setInterestRate(double interestRate) {
   this.interestRate = interestRate;
```

this (first use case)

```
public void setInterestRate(double interestRate) {
    interestRate = interestRate;
}
public void setCurrency(String currency) {
    if (currency.equals("TL") && currency.equals("USD")) {
    balance = balance / 32.88;
}
    if (currency.equals("USD") && currency.equals ("TL")){
    balance = balance * 32.88;
}
    if (currency.equals("TL") || currency.equals("USD")) {
    currency = currency;
}
```

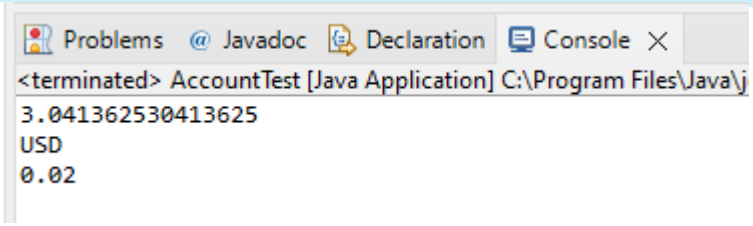
```
public void setInterestRate(double interestRate) {
    this.interestRate = interestRate;
}
public void setCurrency(String currency) {
    if (this.currency.equals("TL") && currency.equals("USD")) {
    balance = balance / 32.88;
}
    if (this.currency.equals("USD") && currency.equals ("TL")){
    balance = balance * 32.88;
}
    if (currency.equals("TL") || currency.equals("USD")) {
    this.currency = currency;
}
```

this (first use case)

```
public void setInterestRate(double interestRate) {
    this.interestRate = interestRate;
}
public void setCurrency(String currency) {
    if (this.currency.equals("TL") && currency.equals("USD")) {
    balance = balance / 32.88;
}
    if (this.currency.equals("USD") && currency.equals ("TL")){
    balance = balance * 32.88;
}
    if (currency.equals("TL") || currency.equals("USD")) {
    this.currency = currency;
}
```

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

    Account account1 = new Account(1, 100, "TL");

    account1.setCurrency("USD");
    System.out.println(account1.getBalance());
    System.out.println(account1.getCurrency());

    account1.setInterestRate(0.02);
    System.out.println(account1.getInterestRate());
}
```

- What should be the output?



this (first use case)

- It can be also used inside constructors to seperate class instances from parameters.

```
// Constructors
public Account(int n, double b, String c){
    number = n;
    if (b > 0)
        balance = b;
    else
        balance = 0;

    interestRate = 0;
    checksetCurrency(c);
}
public Account(int n, String c){
    number = n;
    balance = 0;
    interestRate = 0;

    checksetCurrency(c);
}
public Account(int n){
    number = n;
    balance = 0;
    currency = "TL";
    interestRate = 0;
}
```

First Constructor

```
public Account(int n, double b, String c){
   number = n;
   if (b > 0)
      balance = b;
   else
      balance = 0;

   interestRate = 0;
   checksetCurrency(c);
}
```

```
public Account(int number, double balance, String currency){
   this.number = number;
   if (balance > 0)
      this.balance = balance;
   else
      this.balance = 0;

   this.interestRate = 0;
   this.checksetCurrency(currency);
}
```

Second Constructor

```
public Account(int n, String c){
   number = n;
   balance = 0;
   interestRate = 0;

   checksetCurrency(c);
}
```

```
public Account(int number, String currency){
   this.number = number;
   this.balance = 0;
   this.interestRate = 0;

   checksetCurrency(currency);
}
```

Third Constructor

```
public Account(int n){
   number = n;
   balance = 0;
   currency = "TL";
   interestRate = 0;
}
```

```
public Account(int number){
   this.number = number;
   this.balance = 0;
   this.currency = "TL";
   this.interestRate = 0;
}
```

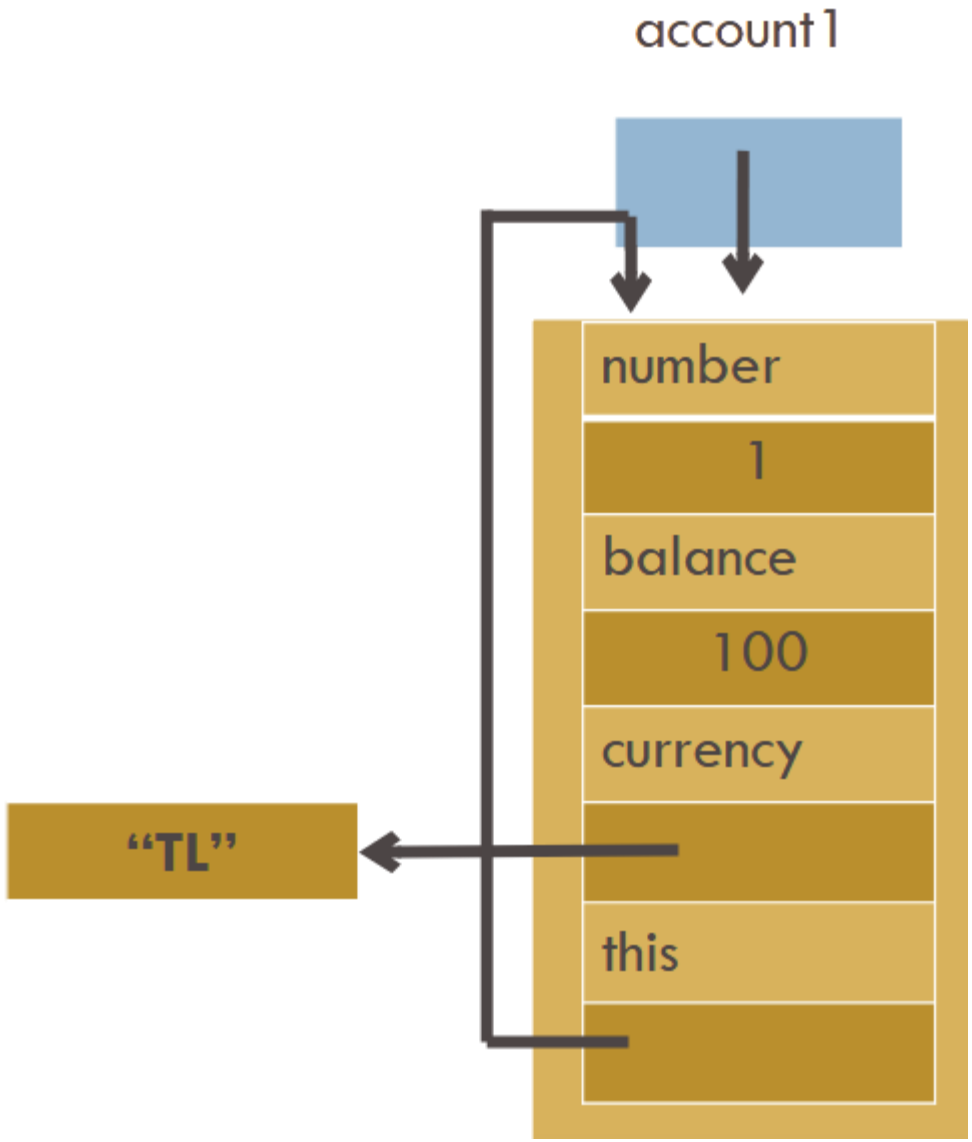
this (first use case)

- To be consistent, the same can be done in **get** methods.

```
public int getNumber() {
   return this.number;
}
public double getBalance() {
   return this.balance;
}
public String getCurrency() {
   return this.currency;
}
public boolean getInterestRate() {
   return this.interestRate;
}
```

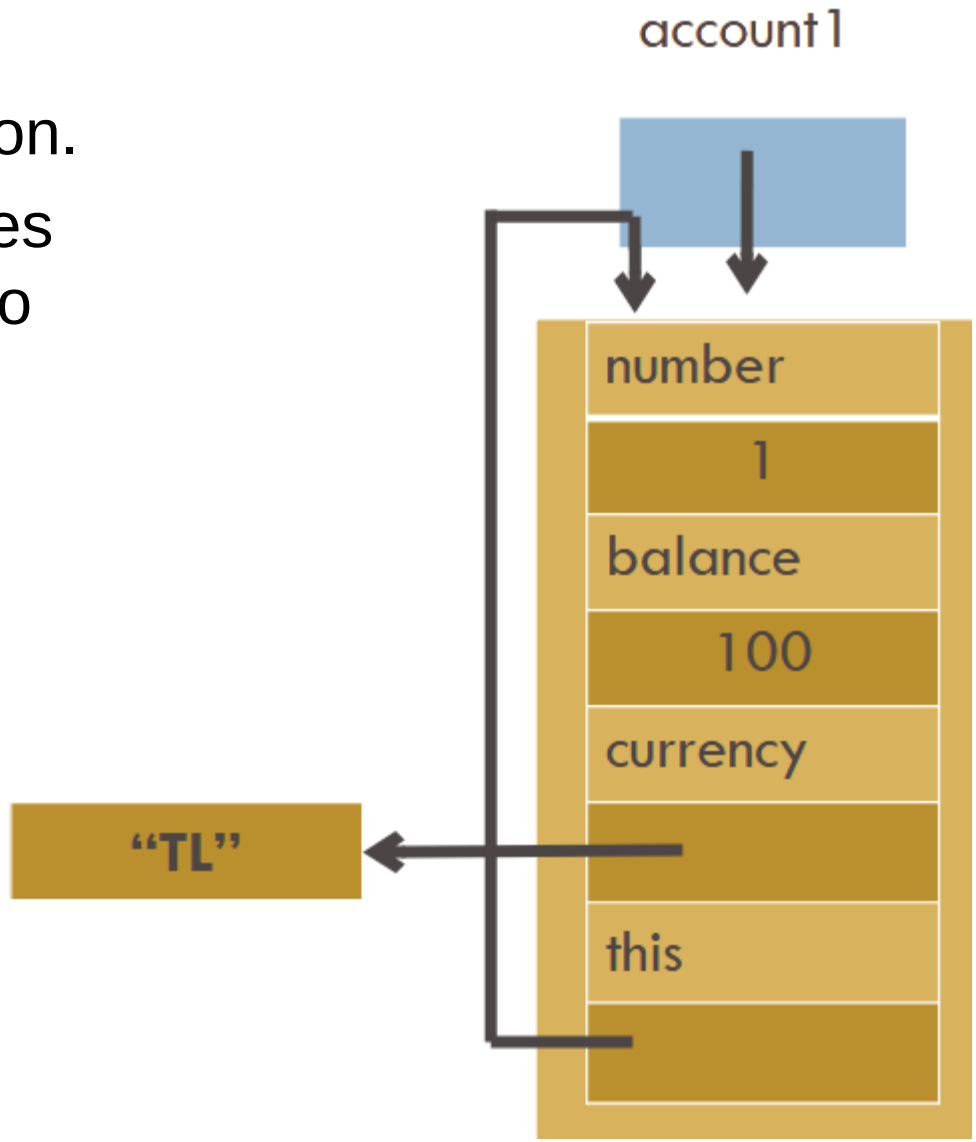
A conceptual model for this

- A conceptual model for understanding ***this***
- Each object in Java has an implicit instance variable, named **this**, that points to the object itself.



A conceptual model for this

- This is not what really happens during execution.
- In Java, each object does NOT keep a reference to itself –
  - –that would be a waste of memory space.
- However, take this as simply a conceptual model that helps us understand what this means.



This (second use case)

- There are several use cases of this
  1. It is used to access the instance variables without any confusion with parameter names.

  2. It can be used to invoke constructors of the same class.

- When you have multiple constructors, ***this*** keyword can be used to call other constructors with a different set of arguments.

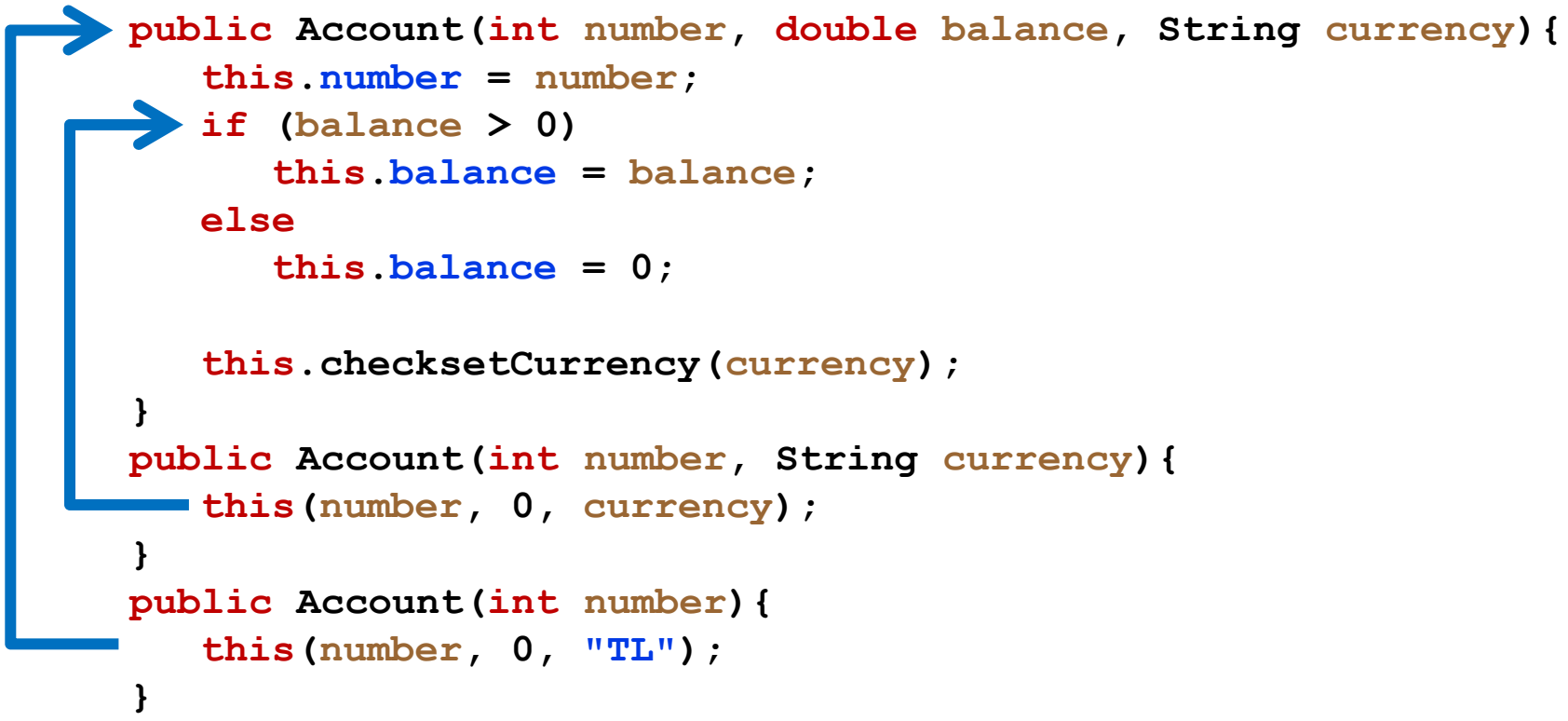
Three Constructors

```
private int number;
private double balance;
private String currency;

// Constructors
public Account(int number, double balance, String currency){
    this.number = number;
    if (balance > 0)
        this.balance = balance;
    else
        this.balance = 0;

    this.checksetCurrency(currency);
}
public Account(int number, String currency){
    this.number = number;
    this.balance = 0;

    checksetCurrency(currency);
}
public Account(int number){
    this.number = number;
    this.balance = 0;
    this.currency = "TL";
}
```

this (second use case)

```
private int number;
private double balance;
private String currency;

// Constructors
public Account(int number, double balance, String currency){
    this.number = number;
    if (balance > 0)
        this.balance = balance;
    else
        this.balance = 0;

    this.checksetCurrency(currency);
}
public Account(int number, String currency){
    this(number, 0, currency);
}
public Account(int number){
    this(number, 0, "TL");
}
```

this (second use case)



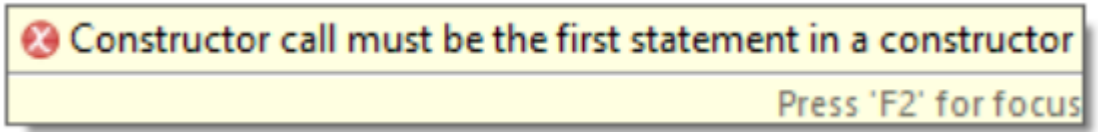
```
private int number;
private double balance;
private String currency;

// Constructors
public Account(int number, double balance, String currency){
   this.number = number;
   if (balance > 0)
      this.balance = balance;
   else
      this.balance = 0;

   this.checksetCurrency(currency);
}
public Account(int number, String currency){
   this(number, 0, currency);
}
public Account(int number){
   this(number, 0, "TL");
}
```

Using this for constructor call

- The constructor call should be the first statement in your constructor.

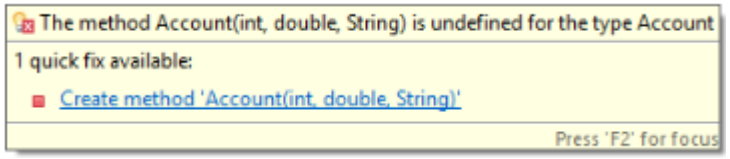
```
public Account(int number, String currency){
      int a;
      this(number, 0, currency);
```

- Otherwise, you will receive the following error!



this

```
public Account(int number, double balance, String currency){
    this.number = number;
    if (balance > 0)
        this.balance = balance;
    else
        this.balance = 0;

    this.checksetCurrency(currency);
}
public Account(int number, String currency){
    this(number, 0, currency);
}
public Account(int number){
    Account(number, 0.0, "TL");
}
```



- You cannot use the constructor of the class inside a constructor to call another constructor.
  - The only way of calling another constructor within a constructor is through using ***this*** keyword.

Any Questions?