

**Document #4: Prepared by: Dr. Khalid A. Darabkeh**  
**Sun Certified Programmer for Java 2 Platform**  
**Oracle Java Developer Certified Trainer**  
**Oracle Database Administrator Certified Associate**

---

*this* is a pointer to the current object. The following is an example of how to use *this*:

***Program#1:***

```
public class This
{
    public static void main(String [] u)
    {
        Test refer1=new Test();
        System.out.println(refer1.method(2,3));// Output = 72
    }
}

class Test
{
    int i=12,j;

    int method(int i,int j)
    {
        i=this.i+(i*j);
        j=i*j-this.j;
        return i+j;
    }
}
```

***Constructors:***

1. Have the same name (with different signature) which is similar to class name
2. Have no return type
3. Are an example of overloading
3. There is a default constructor, which has no signature, for every class if there is no constructor definition.
4. The main job of that default constructor is to initialize state members to its default values ***as long as there is no own definition and initialization***
5. If you define at least a constructor then it becomes your responsibility to define the constructor that has no signature.





char	<b>charAt</b> (int index)
int	<b>compareTo</b> (String anotherString)
int	<b>indexOf</b> (int ch)
int	<b>lastIndexOf</b> (int ch)
int	<b>length</b> ()
String	<b>replace</b> (char oldChar, char newChar)
String	<b>substring</b> (int beginIndex)
String	<b>substring</b> (int beginIndex, int endIndex)  <ol style="list-style-type: none"> <li>1. Index starts with zero</li> <li>2. endIndex is excluded. Thus, the length of the substring is endIndex-beginIndex</li> </ol>
String	<b>toLowerCase</b> ()
String	<b>toUpperCase</b> ()
String	<b>intern</b> ()

*Question: what is the difference between these constructions?*

- ✓ String s1 = "JAVA Programming";
- ✓ String s2 = new String("JAVA Programming ");

**Program#4:**

```
import javax.swing.JOptionPane;

public class StringNotes
{

public static void main(String [] u)
{

String str1="JAVA Programming";
String str2="JAVA Programming";

String str3=new String("JAVA Programming");
String str4=new String("JAVA Programming");

if (str1==str2)
    JOptionPane.showMessageDialog(null,"Both strings are equal");
else
    JOptionPane.showMessageDialog(null,"The strings are not equal");
```

```

if (str3==str4)
    JOptionPane.showMessageDialog(null,"Both strings are equal");
else
    JOptionPane.showMessageDialog(null,"The strings are not equal");

if (str3.intern()==str4.intern())
    JOptionPane.showMessageDialog(null,"Both strings are equal");
else
    JOptionPane.showMessageDialog(null,"The strings are not equal");

if (str3.equals(str4))
    JOptionPane.showMessageDialog(null,"Both strings are equal");
else
    JOptionPane.showMessageDialog(null,"The strings are not equal");

char [] a= {'R','E','G','A','R','D','S'};
String st=String.valueOf(a);
JOptionPane.showMessageDialog(null," After copyValueOf: "+st);

char []a1=new char[16];
a1=str1.toCharArray();
System.out.println(a1[15]); // Output =g

String s1="Alabama";
String s2=s1;
String s3="Birmingham";

System.out.println((int)s1.charAt(0));// Output = 65
System.out.println((int)s3.charAt(0));// Output = 66

int i=s1.compareTo(s3);
int ii=s3.compareTo(s1);
int iii=s1.compareTo(s2);
System.out.println(i); // Output = -1
System.out.println(ii); // Output = 1
System.out.println(iii); // Output = 0

}

}

```

## *Inheritance*

1. It is the key of object-oriented principles
2. Parent class (super class) is the class that is extended by one or more child classes
3. The following terms have the same meaning: parent class, super class, and base class
4. Child class (subclass) is the class that is derived from another class (i.e. super class)
5. The following terms have the same meaning: child class, sub class, and derived class
6. Inheritance is a mechanism at which one class (child class) acquires the properties (data and methods) of another class (i.e. parent class)
7. Getting rid of code duplicates and the easiness of adding fields of methods (i.e. properties) are the key advantages of using inheritance
8. The keyword extends denotes for inheritance
9. There is no multiple-inheritance in Java

### *Method overriding or redefining (deep polymorphism):*

#### **Conditions to be satisfied:**

1. There are two methods. The first one declared in the super class and the other one in the child class.
2. These methods have the same name and return type, do different jobs, and have same signatures (parameters)

#### **Notes:**

1. Don't attempt to use incompatible return type
2. Method (toString()) found inside **Object** class is an example of method overriding.

We are going to study the following entities and then build a model that avoids data redundancy as much as possible (i.e., use inheritance and deep polymorphism):

<b><i>Salary Employee:</i></b>	<b><i>Hourly Employee:</i></b>	<b><i>Commission Employee:</i></b>
First name Last name SSN Age Employee ID Employee type Monthly salary Monthly Tax Yearly income	First name Last name SSN Age Employee ID Employee type Wage/hour Weekly worked hours Monthly Tax Yearly income	First name Last name SSN Age Employee ID Employee type Weekly sales Commission percentage Monthly Tax Yearly income

### ***Program#5: Solution***

```
public class InheritanceDemo
{

    public static void main(String [] ar)
    {

SalaryEmployee ref=new
SalaryEmployee("John","Smith",98765434,1,0.35F,"Manager",50,5000);
System.out.println(ref);

HourlyEmployee ref1=new
HourlyEmployee("Ahmad","Salem",98733455,2,0.25F,"Junior",29,14.5F,48F);
System.out.println(ref1);

CommissionEmployee ref2=new
CommissionEmployee("Loai","Khalid",98233333,3,0.15F,"Freshman (under
probation)",24,5000F,0.1F);
System.out.println(ref2);

    }

}

class Employee
{
String First_Name;
String Last_Name;
long SSN;
int Emp_ID;
float Monthly_Tax;
double yearly_Income(){return 0.0;}
String Employee_Type;
int Age;
}

class SalaryEmployee extends Employee
{

float Monthly_Salary;
SalaryEmployee(String First_Name,String Last_Name,long SSN,int Emp_ID, float
Monthly_Tax,String Employee_Type,int Age, float Monthly_Salary)
{
```

```

this.First_Name=First_Name;
this.Last_Name=Last_Name;
this.SSN=SSN;
this.Emp_ID=Emp_ID;
this.Monthly_Tax=Monthly_Tax;
this.Employee_Type=Employee_Type;
this.Age=Age;
this.Monthly_Salary=Monthly_Salary;
    }

    double yearly_Income()
    {
        double final_income =(Monthly_Salary*12)-
(Monthly_Salary*12*Monthly_Tax);
        return final_income;
    }
    public String toString()
    {
        System.out.println("-----
----");
        return "First Name: "+First_Name+"\n"+"Last Name: "+Last_Name+"\n"+"SSN:
"+SSN+"\n"+"Emp ID: "+Emp_ID+"\n"+"Monthly tax: "+Monthly_Tax+"\n"+"Empolyee
type: "+Employee_Type+"\n"+"Age: "+Age+"\n"+"Monthly salary:
"+Monthly_Salary+"\n"+"Final annual income: "+yearly_Income();
    }

}

class HourlyEmployee extends Employee
{

float Wage_Hour;
float Weekly_Working_Hours;
HourlyEmployee(String First_Name,String Last_Name,long SSN,int Emp_ID, float
Monthly_Tax,String Employee_Type,int Age, float Wage_Hour,float
Weekly_Working_Hours)
{
    this.First_Name=First_Name;
    this.Last_Name=Last_Name;
    this.SSN=SSN;
    this.Emp_ID=Emp_ID;
    this.Monthly_Tax=Monthly_Tax;
    this.Employee_Type=Employee_Type;
    this.Age=Age;
    this.Wage_Hour=Wage_Hour;
    this.Weekly_Working_Hours=Weekly_Working_Hours;
}

```

```

    }

    double yearly_Income()
    {
        double final_income =(Wage_Hour*Weekly_Working_Hours*4*12)-
(Wage_Hour*Weekly_Working_Hours*4*12*Monthly_Tax);
        return final_income;
    }

    public String toString()
    {
        System.out.println("-----
-----");
        return "First Name: "+First_Name+"\n"+"Last Name:
"+Last_Name+"\n"+"SSN: "+SSN+"\n"+"Emp ID: "+Emp_ID+"\n"+"Monthly tax:
"+Monthly_Tax+"\n"+"Empolyee type: "+Employee_Type+"\n"+"Age: "+Age+"\n"+"Wage
per hour: "+Wage_Hour+"\n"+"Weekly working hours:
"+Weekly_Working_Hours+"\n"+"Final annual income: "+yearly_Income();
    }
}

class CommissionEmployee extends Employee
{

    float Weekly_Sales;
    float Commission_Rate;
    CommissionEmployee(String First_Name,String Last_Name,long SSN,int Emp_ID,
float Monthly_Tax,String Employee_Type,int Age, float Weekly_Sales, float
Commission_Rate)
    {
        this.First_Name=First_Name;
        this.Last_Name=Last_Name;
        this.SSN=SSN;
        this.Emp_ID=Emp_ID;
        this.Monthly_Tax=Monthly_Tax;
        this.Employee_Type=Employee_Type;
        this.Age=Age;
        this.Weekly_Sales=Weekly_Sales;
        this.Commission_Rate=Commission_Rate;
    }

    double yearly_Income()
    {
        double final_income =(Weekly_Sales*Commission_Rate*4*12)-
(Weekly_Sales*Commission_Rate*4*12*Monthly_Tax);

```

```

        return final_income;
    }

    public String toString()
    {
        System.out.println("-----
-----");
        return "First Name: "+First_Name+"\n"+"Last Name:
"+Last_Name+"\n"+"SSN: "+SSN+"\n"+"Emp ID: "+Emp_ID+"\n"+"Monthly tax:
"+Monthly_Tax+"\n"+"Empolyee type: "+Employee_Type+"\n"+"Age:
"+Age+"\n"+"Weekly sales: "+Weekly_Sales+"\n"+"Commission rate:
"+Commission_Rate+"\n"+"Final annual income: "+yearly_Income();
    }
}

////////////////////////////////////

```

**Program#6:**

```

public class inheritance
{

public static void main(String []pp)

{
derived pointer = new derived();
pointer.method(5);
int MaxOfThreeNumbers=pointer.FinalMax(3,-5,99);
System.out.println(MaxOfThreeNumbers);

}

}

class base
{

int i=4;
int j=12;
int k=122;
int max(int i,int j)
{
    if (i>j) return i;

```

```

        else return j;
    }
}
class derived extends base
{

    int i=18;
    int p=77;
    int FinalMax(int i, int j, int k)
    {
        int y=max(j,k);
        if (i>y) return i;
        else return y;

    }

    void method(int i)
    {
        System.out.println(i);
        System.out.println(this.i);
        System.out.println(super.i);
    }

}

```

***Output:***

```

5
18
4
99

```

////////////////////////////////////

***Program#7:***

```

public class inheritance
{

    public static void main(String []pp)

    {
        derived pointer = new derived();

        derived pointer1 = new derived("Example of Inheritance" );
    }
}

```

```
derived pointer2 = new derived(5);
System.out.println("pointer2.i =" + pointer2.i);
System.out.println("pointer2.j =" + pointer2.j);
```

```
derived pointer3 = new derived(10,20);
System.out.println("pointer3.i =" + pointer3.i);
System.out.println("pointer3.j =" + pointer3.j);
```

```
}
```

```
}
```

```
class base
```

```
{
```

```
    base()
```

```
    {
```

```
        System.out.println("Hello from Base class");
```

```
    }
```

```
    base(String str)
```

```
    {
```

```
        System.out.println("The string is " + str);
```

```
    }
```

```
}
```

```
class derived extends base
```

```
{
```

```
    int i,j;
```

```
    derived() // Calls base() by default unless you declare super()
```

```
    {
```

```
        System.out.println("Hello from Subclass");
```

```
    }
```

```
    derived(String st) // Calls base() by default unless you declare super()
```

```
    {
```

```
        System.out.println(st);
```

```
    }
```

```
    derived(int i) // Calls base() by default
```

```
    {
```

```
        this.i=i;
```

```
    }
```



```

}

}
class Test1

{
    int []a = new int [6];
    void method ()
    {
        for (int i =0; i<a.length; i++)
        {
a[i]=Integer.parseInt(JOptionPane.showInputDialog("Enter Array of 6 Elements"));
        }
    }
}

class Test2 extends Test1

{
    int num;

    Test2(int f)
    {
        num=f;
    }

    void method ()
    {
        super.method();
        int count=0;

        for (int i =0; i<a.length; i++)
        {
            if (a[i]== num)
                count++;
        }

        if (count>=1)
            System.out.println(" The number was found "+count+" times");
        else
            System.out.println(" The number was not found");
    }
}

```

////////////////////////////////////

## *public class Object*

Class Object is the default parent class (i.e. every class is a child, subclass, of class Object). It is the root of class hierarchy. Thus, any class including arrays can access or implement Object's methods.

<i>RDT</i>	<i>Important Object Methods</i>
boolean	<b>equals</b> ( Object obj)
String	<b>toString</b> ()
protected void	<b>finalize</b> () called by garbage collector when decision is made that the object is no longer in use (i.e. there are no pointers or references to that object)

### *Program#9:*

```
public class toString
{

public static void main(String [] u)
{

Test1 refer1=new Test1();
Test2 refer2=new Test2();

System.out.println(refer1);
System.out.println(refer2);

}

}

class Test1
{

public String toString()
{
return "This is a reference to an object of type Test1";
}

}
```

```
class Test2 {}
```

***Output:***

```
This is a reference to an object of type Test1
Test2@a90653
```

////////////////////////////////////

***public final class Math extends Object***

<i>DT</i>	<i>Math Data</i>
static double	<b>E</b>
static double	<b>PI</b>

<i>RDT</i>	<i>Important Math Methods</i>
static double	<b>abs</b> (double a)
static float	<b>abs</b> (float a)
static int	<b>abs</b> (int a)
static long	<b>abs</b> (long a)
static double	<i>ceil</i> (double a)
static double	<b>floor</b> (double a)
static double	<b>cos</b> (double a)
static double	<b>sin</b> (double a)
static double	<b>tan</b> (double a)
static double	<i>cosh</i> (double x)
static double	<i>sinh</i> (double x)
static double	<b>sqrt</b> (double a)
static double	<b>exp</b> (double a)
static double	<b>log</b> (double a)
static double	<i>log10</i> (double a)
static double	<b>max</b> (double a, double b)
static float	<b>max</b> (float a, float b)
static int	<b>max</b> (int a, int b)
static long	<b>max</b> (long a, long b)
static double	<b>min</b> (double a, double b)
static float	<b>min</b> (float a, float b)
static int	<b>min</b> (int a, int b)
static long	<b>min</b> (long a, long b)

static double	<b>pow</b> (double a, double b)
static double	<b>random</b> () (Returns a double value with a positive sign, greater than or equal to 0.0 and less than 1.0)

***Program#10:***

```

public class Notes
{

public static void main(String [] a)
{

SetandGetDemo pointer = new SetandGetDemo();

// Use Set Methods
pointer.setDay(30);
pointer.setMonth(10);
pointer.setYear(2007);

// Use Get Methods
System.out.println(pointer.getDay());
System.out.println(pointer.getMonth());
System.out.println(pointer.getYear());

System.out.println(pointer);

Chain ref=new Chain();
ref.setDay(30).setMonth(10).setYear(2007);

System.out.println(ref.getEverything());
}
}

class SetandGetDemo
{

int day;
int month;
int year;

void setDay(int day)
{
    this.day=day;
}
}

```

```

    }
void setMonth(int month)
    {
        this.month=month;
    }
void setYear(int year)
    {
        this.year=year;
    }

int getDay()
    {
        return day;
    }
int getMonth()
    {
        return month;
    }
int getYear()
    {
        return year;
    }

public String toString()
    {
        return "The date is "+day+" /"+month+" /"+year;
    }
}

```

```

class Chain
{

int day;
int month;
int year;

Chain setDay(int day)
    {
        this.day=day;
        return this;
    }
Chain setMonth(int month)
    {

```

```
        this.month=month;
        return this;
    }
void setYear(int year)
    {
        this.year=year;
    }

String getEverything()
    {
        return "The date is "+day+" /"+month+" /"+year;

    }

}
```

***Output:***

30

10

2007

The date is 30 /10 /2007

The date is 30 /10 /2007