Objectives

- In this session, you will learn to:
  - Use logical operators
  - Iterate with loops
  - Create arrays
  - Use switch branching statements
  - Use Java classes, methods, and constructors
  - Use package and import statements
  - Work with pass-by-value concepts

Logical Operators

- Java provides the following logical operators:
  - Equality and relational operators:
    - `==` Equal to
    - `!=` Not equal to
    - `>` Greater than
    - `>=` Greater than or equal to
    - `<` Less than
    - `<=` Less than or equal to
  - Conditional operators:
    - `&&` Conditional-AND
    - `||` Conditional-OR
    - `?:` Ternary (shorthand for `if-then-else` statement)
  - Type comparison operator:
    - `Instanceof`

Loop

- Loops:
  - Help to execute a block of code repeatedly.
  - Gets executed for a specific number of iterations or until the condition evaluates to false.
  - Types:
    - `for`
    - `while`
    - `do-while`
    - `For-each`

for Loop

- The `for` loop:
  - Is used to iterate for a fixed number of times.
  - Consists of the following three expressions separated by a semicolon:
    - The initialization expression
    - The test expression
    - The iteration expression (increment/decrement)

while and do-while Loop

- The `while` loop performs a test and continues if expression evaluates to true.
- In the `do-while` loop, the condition test is performed after the expression has run at least once.

For each Loop(also called the "enhanced for **loop**”**)**

- It starts with the keyword for like a normal for-**loop**. Instead of declaring and initializing a **loop** counter variable.
- declare a variable that is the same type as the base type of the array, followed by a colon, which is then followed by the array name.

Arrays and for-each Loop

- Array:
  - Group of variables of the same data type
  - Referred by a common name
  - Created as an object by default
  - Code snippet to declare and initialize an array:

```
String[]names;
names = new String[3];
```

- The `for-each` loop is used to traverse each element in an array, even if the length is unknown.

Java Naming Conventions

- In Java:
  - The class names should be nouns in upper camel case.
  - Methods should be verbs in lower camel case.
  - Variable names should be short but meaningful.
  - One-character variable names should be avoided except as temporary variables.
  - Constants should be declared using all uppercase letters.

Methods

- Methods:
  - Are created to manipulate data fields of a class.
  - Can be used to set the value of each field.
  - Can be used to retrieve the value of each field

Constructors

- Constructor:
  - Used to create an instance of a class
  - Can take parameters
  - Without arguments is called a `no-arg` constructor
- The following code snippet shows a `no-arg` constructor:

```
public class Employee
{
                public Employee()
  {
}
}
```

- The following code snippet shows how the constructor is implicitly invoked:

```
Employee emp = new Employee();
```

Creating an Instance of an Object

- `new` keyword:
  - Used to create an instance of a class
- Example:

```
 Employee emp = new Employee();
  emp.empId = 101;
// legal if the field is public, but
not good                        // OO practice
  emp.setEmpId(101); // use a method
instead
  emp.setName("John Smith");
  emp.setSsn("011-22-3467");
  emp.setSalary(120345.27);
```

package Statement

- Package:
  - Declared using the `package` keyword
  - Used to group Java classes
  - Implemented as a folder
  - Provides a namespace to a class
  - Declaration must always appear at the top of the file

More on import

- The `import` statement:
  - Follows the package declaration and precede the class declaration.
  - Is not mandatory for an application.
- By default, a class always imports `java.lang.*;`.
- There is no need to import classes that are in the same package.

Java Is Pass-By-Value

- Java uses pass-by-value for all assignment operations.

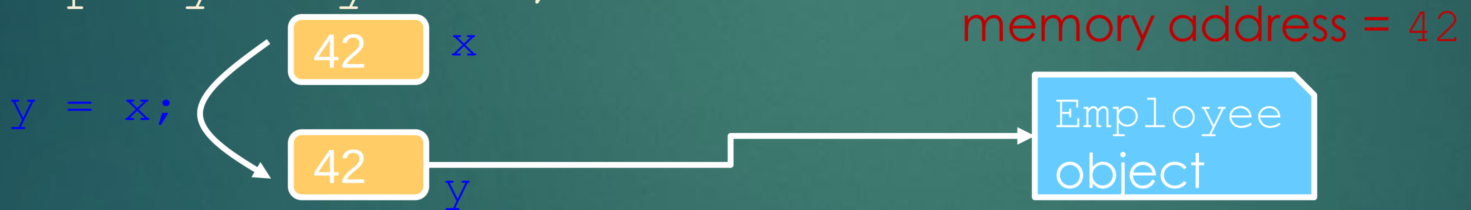
```
    into x = 3;
    into y = x;
```

- If `x` is later modified (for example, `x = 5;`), the value of `y` remains unchanged.

Pass-By-Value for Object References

- For Java objects, the value of the right side assignment is a reference to memory that stores the object.
- Example:

```
Employee x = new Employee();
Employee y = x;
```



Here, the value of `y` is equivalent of `x`. Therefore, both `x` and y hold the reference to the same `Employee` object.

Objects Passed as Parameters

- Consider the following code snippet to understand the concept of memory allocation for an object:

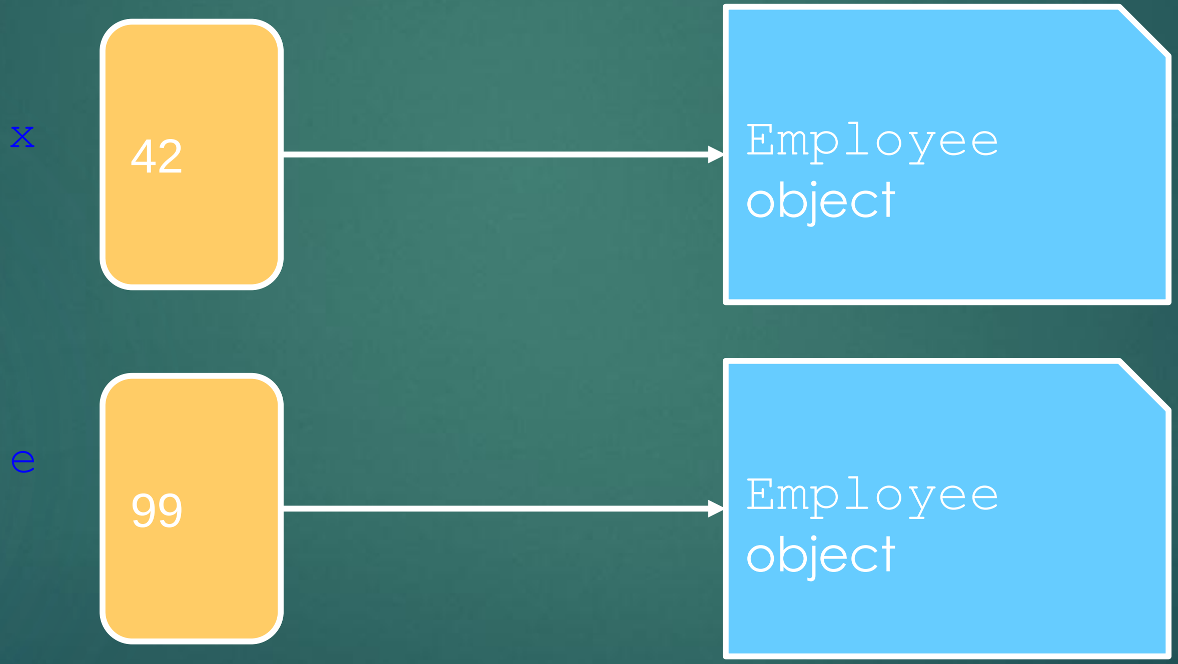
```
Employee x = new Employee();

foo(x);

public void foo(Employee e)
{
    e = new Employee();
    e.setSalary (1_000_000.00);
// What happens to x here?
}
```

Objects Passed as Parameters (Contd.)

- The value of `x` is unchanged even after a call to the `foo()` method, as shown in the following figure.


x
42
Employee object
e
99
Employee object