

# Introduction to Programming

*with Java, for Beginners*

Private  
Static vs. Dynamic

## Asking Object about its data directly

- It *may* be possible to ask a object about its data without querying the object
  - public or no modifier
  - *ObjectName.DataField;*
- But you can prevent such change by making object data *private*
  - E.g. *private int age;*

Example in Dr Java  
> Student s1 = new  
Student("Joe",5);  
> s1.age  
5  
> s1.age = 6;  
> s1.age  
6

ESE112

2

## Encapsulation or Information Hiding

- One of the advantages of OOP is that object need not reveal all of its attributes (data/state) and behavior
- We can hide details of one object from another
- Use modifiers (private/public) to hide information
  - Ideally we make all instance variable(s) private
- Provide methods (query/command) if you want to allow the data to read or written
  - Getter methods to read e.g. getAge()
  - Setter methods to modify e.g. setAge() -> not necessary to provide

ESE112

3

## Dynamic Variables and Methods

- All instance variables (object data) and methods (object behavior) created without static keyword
  - Note: There is no "dynamic" keyword in Java
  - Dynamic by default
- In general, *dynamic* refers to things created at "run time" i.e. when the program is running
- Every object gets its own (dynamic) instance variables
- Every object effectively gets its own copy of each dynamic method (i.e. the instructions in the method)

ESE112

4

## Static Variables with OO class

- *Static* means “pertaining to the class in general”, *not* to an individual object
- Variable is declared with the *static* keyword outside all methods
- A static variable is *shared* by all instances (if any)
  - All instances may be able read/write it

ESE112

5

## Use of static variable I

- Global Constants
  - Constants are variable that don't change
  - Constants are made static because there is no need for more than one copy it
- Example:

```
class Deck{
    public static final int JACK = 11;
    public static final int QUEEN = 12;
    public static final int KING = 13;
    public static final int SPADE = 1;
    ....
}
```

ESE112

6

## Use of static variable II

- Providing communication among instances of classes i.e. objects
- In this case using static variable is way of accessing some common resource

ESE112

7

## Example: Ticket No. Generator

```
public class Ticket{
    // shared
    private static int numTicketsSold = 0;

    // one per object
    private int ticketNum;

    public Ticket(){
        numTicketsSold = numTicketsSold + 1;
        ticketNum = numTicketsSold;
    }
}
```

**Note:**static variable is used to generate ticketNum and in way keeps track of the number of tickets sold which can be accessed by all objects

ESE112

8

## Static Methods with OO class

- A method may be declared with the *static* keyword
- Static methods live at *class level*, not at *object level*
- Static methods can *access* static variables and other methods, but not dynamic ones
  - How could it? We have not created any objects yet, so it not know who's data we are trying to access.
- Example:

```
public static int getNumSold(){
    return numTicketsSold;
}
```

ESE112

9

## Example: Ticket No. Generator

```
public class Ticket{
    private static int numTicketsSold = 0; // shared
    private int ticketNum; // one per object

    public Ticket(){
        numTicketsSold = numTicketsSold + 1;
        ticketNum = numTicketsSold;
    }

    public static int getNumberSold() {
        return numTicketsSold;
    }

    public int getTicketNumber() {
        return ticketNum;
    }

    public String getInfo(){
        return "ticket # " + ticketNum + "; " +
            numTicketsSold + " ticket(s) sold.";
    }
}
```

ESE112

10

## Static Variables & Methods in General

- A static method that is public can be accessed outside class definition
  - *ClassName.methodName(args)*
  - `double result = Math.sqrt(25.0);`
  - `int sold = Ticket.getNumberSold();`
- A static variable that is public may be accessed
  - Using *ClassName.variableName*
    - E.g. `Math.PI`, `Math.E`
  - Static variables act as global variable i.e. accessible within any static method

ESE112

11

## When to use static with OOP

- A variable should be static if
  - It logically describes the class as a whole
  - There should be only one copy of it
- A method should be static if:
  - It does not use or affect the object that receives the message (it uses only its parameters)

ESE112

12

## Static & Dynamic Rules Recap

- *static* variables and methods belong to the class in general, not to individual objects
- The *absence* of the keyword *static* before non-local variables and methods means *dynamic* (one per object/instance)
- A dynamic method can access all dynamic *and* static variables and methods in the same class
- A static method can not access a dynamic variable (*How could it choose or which one?*)
- A static method can not call a dynamic method (*because dynamic method might access an instance variable*)