

# Introduction to Programming

*with Java, for Beginners*

Polymorphism  
- Overloading  
- Overriding

## Polymorphism

- Polymorphism means *many* (poly) *shapes* (morph)
- In Java, **polymorphism** refers to the fact that you can have multiple methods with the same name in the same class
- There are two kinds of polymorphism:
  - **Overloading**
    - Two or more methods with *different signatures*
  - **Overriding**
    - A method in a subclass to “override” a method in the superclass that has the *same signature*
- We’ve already seen Overloading scenario with Constructors  
E.g. `public AnyLength() {..}`  
`public AnyLength(int n) {..}`

ESE112

1

## Method Overloading

Method *overloading* occurs when

- A class has two or more methods with the same *name* but *different signatures*
  - Different signature -> the number, order, or types of their parameters differ

```
// the foo method is overloaded
public void foo() {.. }
public void foo(int x) { ..}
public void foo(double x){..}
public void foo(int x, double y) {..}
```

- When the `foo(..)` method is called, Java picks the one that “matches”. E.g.

```
foo(10, 350.5);
```

ESE112

2

## Overriding

- *Overriding* occurs if
  - There are two or more methods with the same name *and the same signature* in an inheritance chain
  - For example, the Object class has a `toString()` method
    - It can be *overridden* in a subclass simply by creating a method with the same signature  
`public String toString() {..}`
  - Java picks the “lowest” method in the inheritance chain possible

ESE112

3

## Overriding Variables

- You can, but you shouldn't
- Possible for child class to declare variable with same name as variable inherited from parent class
  - One in child class is called **shadow variable**
    - It shadows the variable with same name in the parent class
  - Confuses everyone!
- Child class already can gain access to inherited variable (provided there are protected) with same name
  - There's no good reason to declare new variable with the same name