# Inheritance

| | |
|---|---|
| **Inheritance** | process of deriving a new class from an existing one. |

## Parent Class vs Child Class

### Parent Class

| | |
|---|---|
| **Parent Class** | super class |
| | base class |

```
1  public class Parent {}
```

- Can't access the methods/data from child class (private or public)
- If not specified it extends Object class, contained in java.lang
- May have many subclasses (Child classes)

### Child Class

| | |
|---|---|
| **Child Class** | class that "extends" a super class |
| | sub class |
| | derived class |
| **Single inheritance** | a single class cannot extend multiple parents, however there can be a chain |

```
1  public class Child extends Parent {}
```

- Child class can only extend one Parent class (Single inheritance)
- Subclasses inherit the interface and implementation of their superclass
- Can access public data/methods of parent

- Contains parents private data/methods, even though it can't call it
  - May "get" or set private data using public methods

```
1  public String getName()
2   return name;
```

- Can call parent methods without qualification, like they are from child class (Instance data)
- Child classes follow an **is-a** relationship (hierarchy)
  - A human is-a Mammal
- The subclass can also:
  - Hide
    - (write a new methods/variables with the same name/signature)
  - Overload
  - Override

---

## Example of Parent and Child

#### Parent Class

```
1  public class Vehicle {
2    protected int wheels;
3    protected void go()
4  }
```

#### Child Class

```
1  public class Car extends Vehicle {
2    private void drive() {
3      int x = wheels;
4      go();
5    }
6  }
```

---

# Access Modifiers

| Modifier | Class | Package | Subclass | World |
|---|---|---|---|---|
| public | Y | Y | Y | Y |
| protected | Y | Y | Y | N |
| default | Y | Y | N | N |
| private | Y | N | N | N |

> The default (no) modifier is also called “package private.”

---

# Super

| | |
|---|---|
| **super** | access to parents members |
| | “this()” for parent class |

- Common use: To invoke a parent’s constructor
  - Has to be first
  - Can use constructors with private parameters/data
  - Example:

```
1  super(instanceVariable)
```

- Can be used to invoke parent methods via dot-access
  - Example:

```
1  super.parentMethod();
```

- Can be used to access immediate parent class instance variable
  - Example

```
1  super.parentInstanceVariable
```

- super() will be inserted if is not stated
  - If a super class doesn’t include a no-args, then the subclass constructors must include a super call
- Constructors are finished from “Top to bottom”
  - Parent class constructors are finished first

---

## Example of Super

### Superclass

```
public class Person {
  private String name;

  public Person(String name) {
    this.name = name;
  }

  public String getName() {
    return name;
  }
}
```

### Subclasses

```
public class Student {
  protected int year;

  public Student(String name, int year) {
    super(name) //Can get the data, using getName()
    this.year = year;
  }

  public Student(String name) {
    this(name, -1); //Can use Constructor chaining
  }
}
```

```
public class Dentist extends Person {
  protected string degree;

  public Dentist(String name) {
    super(name);
    /*If not explixitaly stated in the first line of constructor,
    Java will insert super().
    Exception: If you have a chained call to another
    constructor in same class
    */
    degree = "DDS";
  }
}
```

---

# Glossary

| | |
|---|---|
| **Child Class** | class that “extends” a super class |
| | sub class |
| | derived class |
| **Inheritance** | process of deriving a new class from an existing one. |
| **Parent Class** | super class |
| | base class |
| **Single inheritance** | a single class cannot extend multiple parents, however there can be a chain |
| **super** | access to parents members |
| | “this()” for parent class |