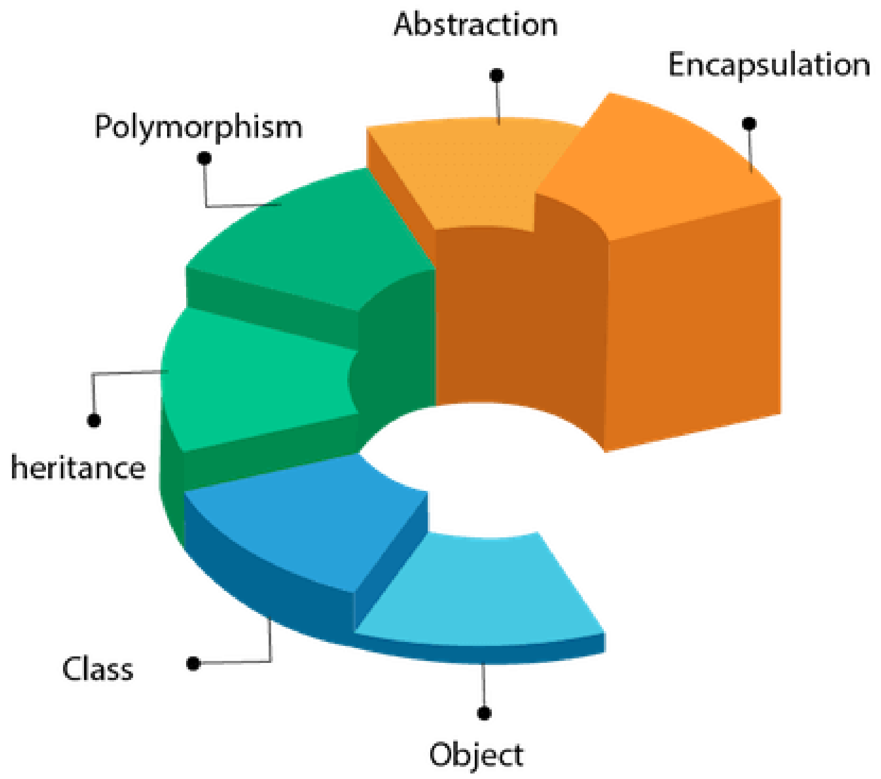
Inheritance

CSX3002/ITX2001 Object-Oriented Concepts and Programming
CS4402 Selected Topic in Object-Oriented Concepts
IT2371 Object-Oriented Programming

OOP

- Class
- Object
- Inheritance
- Polymorphism
- Abstraction
- Encapsulation



Inheritance

- A mechanism in which one object acquires all the properties and behaviors of a parent object.
- New classes is possible to construct from the existing classes
- Besides derive/inherit fields and/or methods from parent class, new method(s) and field(s) can be added to the current class.

Terms used in Inheritance

- Class
- Subclass/ Child Class
- Superclass/ Parent Class
- Reuseablility
- Overiding

The syntax of Java Inheritance

```
class Subclass-name extends Superclass-name  {
   //methods and fields
}
```

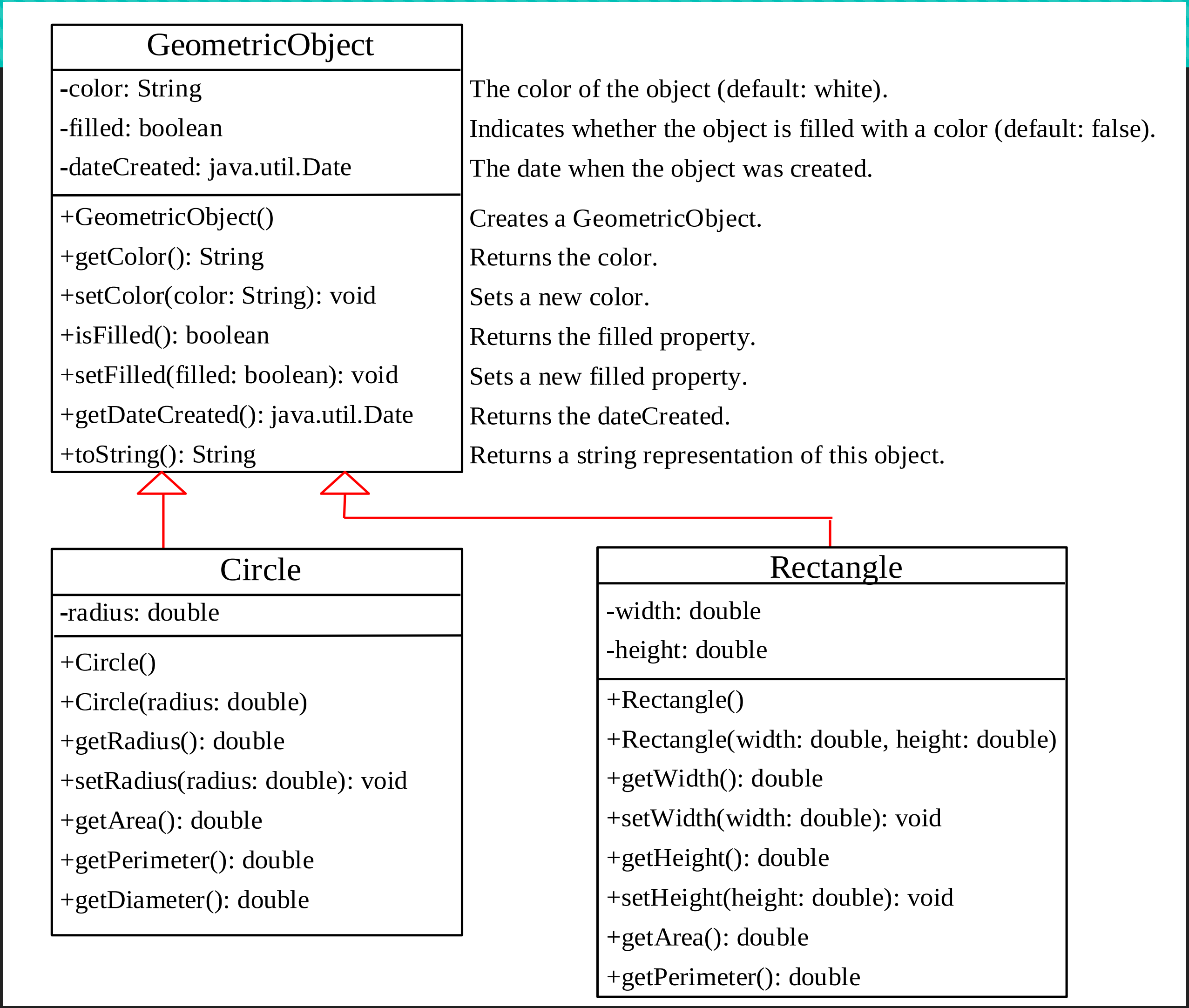
Superclass and subclasses

GeometricObject

Circle

Rectangle

TestCircleRectangle

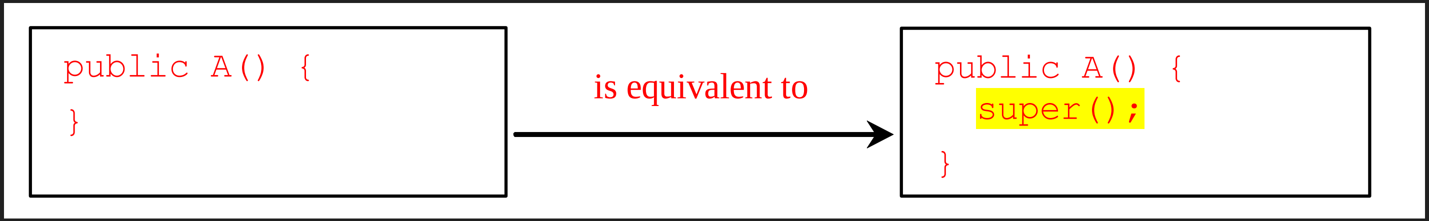

GeometricObject
-color: String
-filled: boolean
-dateCreated: java.util.Date
+GeometricObject()
+getColor(): String
+setColor(color: String): void
+isFilled(): boolean
+setFilled(filled: boolean): void
+getDateCreated(): java.util.Date
+toString(): String
The color of the object (default: white).
Indicates whether the object is filled with a color (default: false).
The date when the object was created.
Creates a GeometricObject.
Returns the color.
Sets a new color.
Returns the filled property.
Sets a new filled property.
Returns the dateCreated.
Returns a string representation of this object.
Circle
-radius: double
+Circle()
+Circle(radius: double)
+getRadius(): double
+setRadius(radius: double): void
+getArea(): double
+getPerimeter(): double
+getDiameter(): double
Rectangle
-width: double
-height: double
+Rectangle()
+Rectangle(width: double, height: double)
+getWidth(): double
+setWidth(width: double): void
+getHeight(): double
+setHeight(height: double): void
+getArea(): double
+getPerimeter(): double

Which part of superclass are Inherited?

- Unlike properties and methods, a superclass's constructors are not inherited in the subclass.
- A constructor is used to construct an instance of a class.
    - They are invoked explicitly or implicitly.
    - They can only be invoked from the subclasses' constructors, using the keyword super.
    - *If the keyword super is not explicitly used, the superclass's no-arg constructor is automatically invoked.*

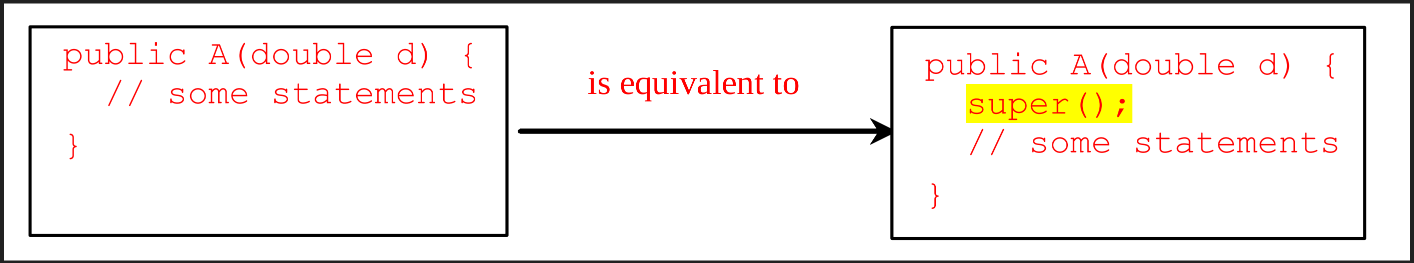
Superclass's Constructor Is Always Invoked

- A constructor may invoke its overloaded constructor or its superclass's constructor.
- If none of them is invoked explicitly, the compiler puts super() as the first statement in the constructor.



```
public A() {
}
```

```
public A() {
  super();
}
```



```
public A(double d) {
  // some statements
}
```

```
public A(double d) {
  super();
  // some statements
}
```

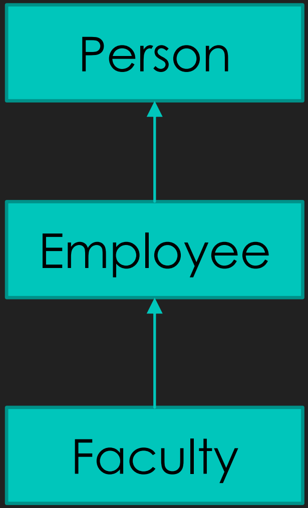
Using the Keyword `super`

- The keyword `super` refers to the superclass of the class in which `super` appears. This keyword can be used in two ways:

1. To call a superclass constructor
2. To call a superclass method

Constructor Chaining

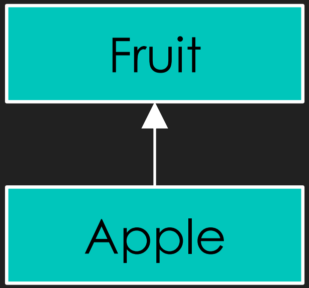
- Constructing an instance of a class invokes all the super classes' constructors along the inheritance chain. This is called *constructor chaining*.



```java
public class Faculty extends Employee {
  public static void main(String[] args) {
    new Faculty();
  }

  public Faculty() {
    System.out.println("(4) Faculty's no-arg constructor is invoked");
  }
}

class Employee extends Person {
  public Employee() {
    this("(2) Invoke Employee’s overloaded constructor");
    System.out.println("(3) Employee's no-arg constructor is invoked");
  }

  public Employee(String s) {
    System.out.println(s);
  }
}

class Person {
  public Person() {
    System.out.println("(1) Person's no-arg constructor is invoked");
  }
}
```

Can you locate error in the program?

```
public class Apple extends Fruit {
}

class Fruit {
  public Fruit(String name) {
    System.out.println("Fruit's constructor is invoked");
  }
}
```



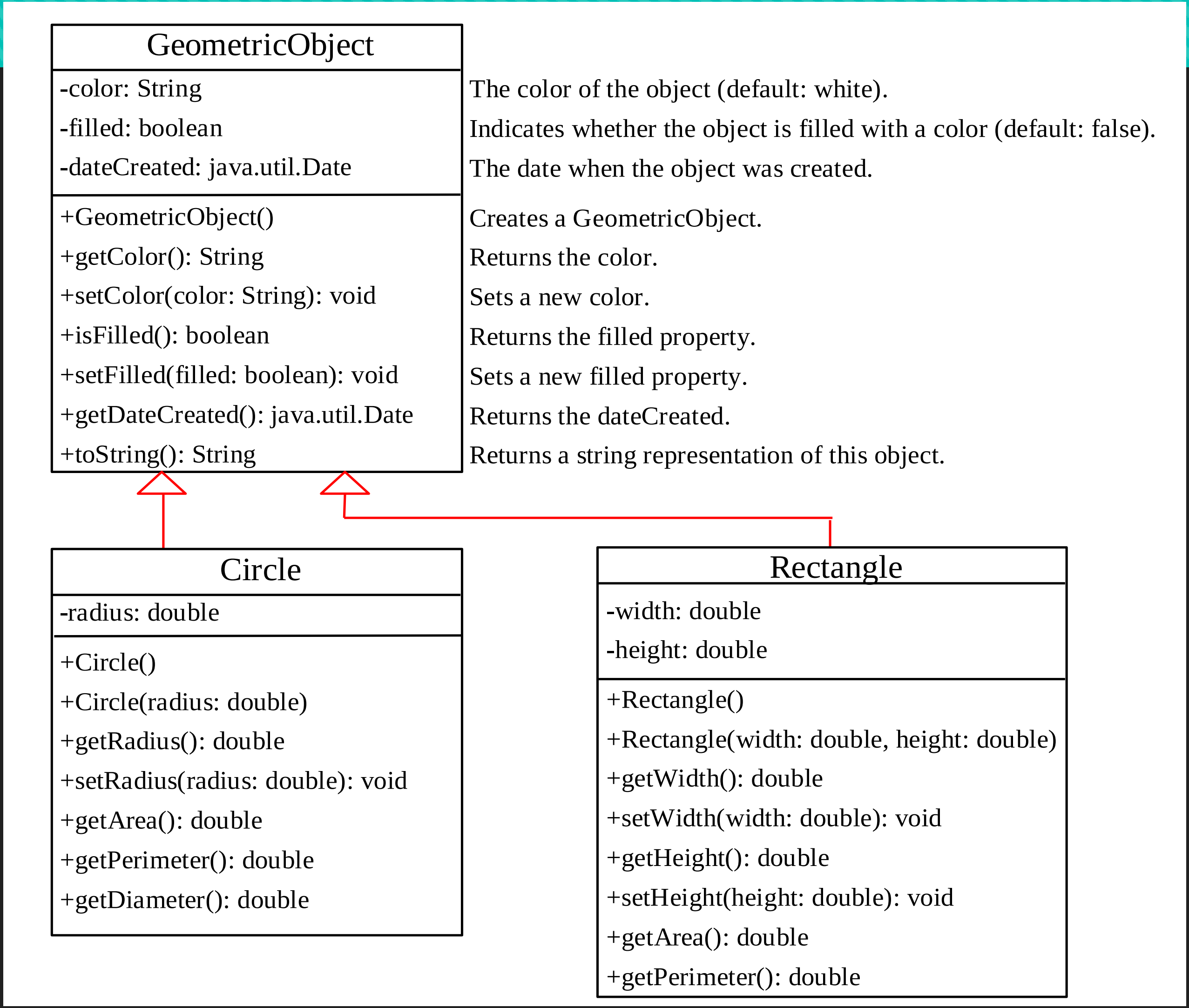
Superclass and subclasses

GeometricObject

Circle

Rectangle

TestCircleRectangle



GeometricObject

-color: String
-filled: boolean
-dateCreated: java.util.Date

+GeometricObject()
+getColor(): String
+setColor(color: String): void
+isFilled(): boolean
+setFilled(filled: boolean): void
+getDateCreated(): java.util.Date
+toString(): String

The color of the object (default: white).
Indicates whether the object is filled with a color (default: false).
The date when the object was created.
Creates a GeometricObject.
Returns the color.
Sets a new color.
Returns the filled property.
Sets a new filled property.
Returns the dateCreated.
Returns a string representation of this object.

Circle

-radius: double

+Circle()
+Circle(radius: double)
+getRadius(): double
+setRadius(radius: double): void
+getArea(): double
+getPerimeter(): double
+getDiameter(): double

Rectangle

-width: double
-height: double

+Rectangle()
+Rectangle(width: double, height: double)
+getWidth(): double
+setWidth(width: double): void
+getHeight(): double
+setHeight(height: double): void
+getArea(): double
+getPerimeter(): double

Calling Superclass Methods

You could rewrite the printCircle() method in the Circle class as follows:

```
public void printCircle() {
  System.out.println("The circle is created " +
    super.getDateCreated() + " and the radius is " + radius);
}
```

Overriding Superclass Methods

- A subclass inherits methods from a superclass.
- Sometimes it is necessary for the subclass to modify the implementation of a method defined in the superclass.
- This is referred to as *method overriding*.

```
public class Circle extends GeometricObject {
  // Other methods are omitted


  /** Override the toString method defined in GeometricObject */
  public String toString() {
    return super.toString() + "\nradius is " + radius;
  }
}
```

NOTE

- An instance method can be overridden only if it is accessible.
- A private method cannot be overridden, because it is not accessible outside its own class.
- If a method defined in a subclass is private in its superclass, the two methods are completely unrelated.

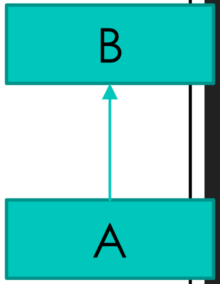
NOTE (cont.)

- Like an instance method, a static method can be inherited.
- However, a static method cannot be overridden.
- If a static method defined in the superclass is redefined in a subclass, the method defined in the superclass is hidden.

Overriding & Overloading

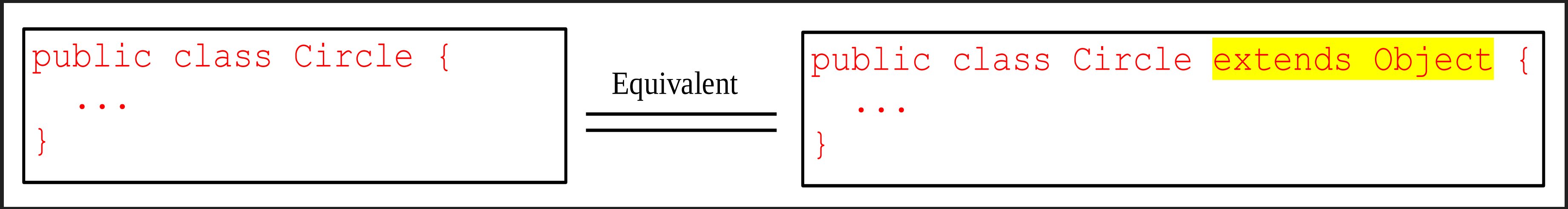
```
public class Test {
  public static void main(String[] args) {
    A a = new A();
    a.p(10);
  }
}

class B {
  public void p(int i) {
  }
}

class A extends B {
  // This method overrides the method in B
  public void p(int i) {
    System.out.println(i);
  }
}
```

```
public class Test {
  public static void main(String[] args) {
    A a = new A();
    a.p(10);
  }
}

class B {
  public void p(int i) {
  }
}

class A extends B {
  // This method overloads the method in B
  public void p(double i) {
    System.out.println(i);
  }
}
```



The Object Class

- Every class in Java is descended from the java.lang.Object class.
- If no inheritance is specified when a class is defined, the superclass of the class is Object.



```
public class Circle {
  ...
}
```

Equivalent

```
public class Circle extends Object {
  ...
}
```

The toString() method in Object

- The toString() method returns a string representation of the object.
- The default implementation returns a string consisting of a class name of which the object is an instance, the at sign (@), and a number representing this object.

```
Loan loan = new Loan();
System.out.println(loan.toString());
```

The toString() method in Object (cont.)

```
Loan loan = new Loan();
System.out.println(loan.toString());
```

- The code displays something like Loan@15037e5 .
- This message is not very helpful or informative.
- Usually, you should override the toString method so that it returns a digestible string representation of the object.

Reference

- Liang, “Introduction to Java Programming”, 6th Edition, Pearson Education, Inc.
- https://www.javatpoint.com/inheritance-in-java