Object Oriented Analysis and Design using Java - UE19CS353

Prof. Sindhu R Pai
Department of Computer Science and Engineering

Object Oriented Analysis and Design using Java

Abstract class and Object class

Prof. Sindhu R Pai

Department of Computer Science and Engineering

Object Oriented Analysis and Design using Java

Abstract class and Object class - Agenda

1. Introduction to Abstract class
2. Creation and Usage in Java
3. Coding examples – Demo
4. Introduction to Object class
5. Methods in Object class
6. Coding examples

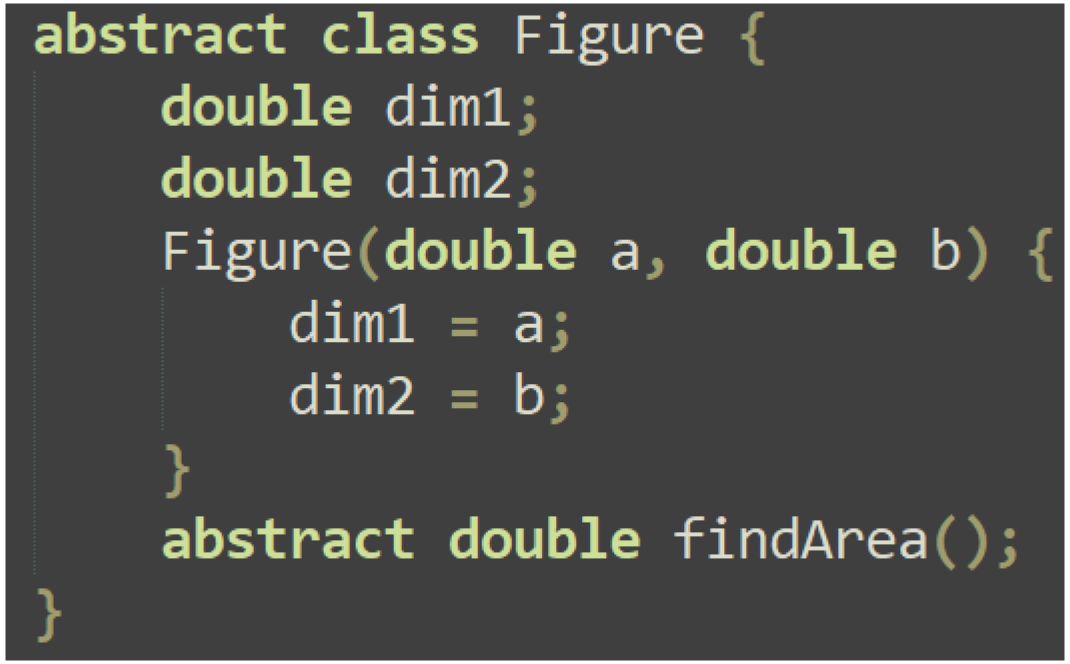
Introduction

- Provides **implementation reuse – provides default implementation**
- To create a super class that only defines a generalized form that will be shared by all of its subclasses, leaving it to each subclass to fill in the details.
- The super class determines the nature of methods that the subclasses must implement.
- Referred to as **subclasser responsibility** because they have no implementation specified in the superclass.
- No method body is present.

Creation and usage in Java

- Created using **abstract** keyword at the beginning of the class declaration.
- May contain abstract methods, i.e., methods without body
- If a class has **at least one abstract method**, then the **class must be declared abstract**
- **Cannot be instantiated**
- If a class extends abstract class then either it has to provide implementation of all abstract methods or declare this class as abstract class
- Can have both **static and non-static data members and methods** like any other java class
- **Can not be final** in Java because abstract classes are used only by extending
- A class **can extend only one abstract class** as Java does not support multiple inheritance

Coding Example: If the abstract class contains the below data, how to implement Rectangle and Triangle classes?



```
abstract class Figure {
    double dim1;
    double dim2;
    Figure(double a, double b) {
        dim1 = a;
        dim2 = b;
    }
    abstract double findArea();
}
```

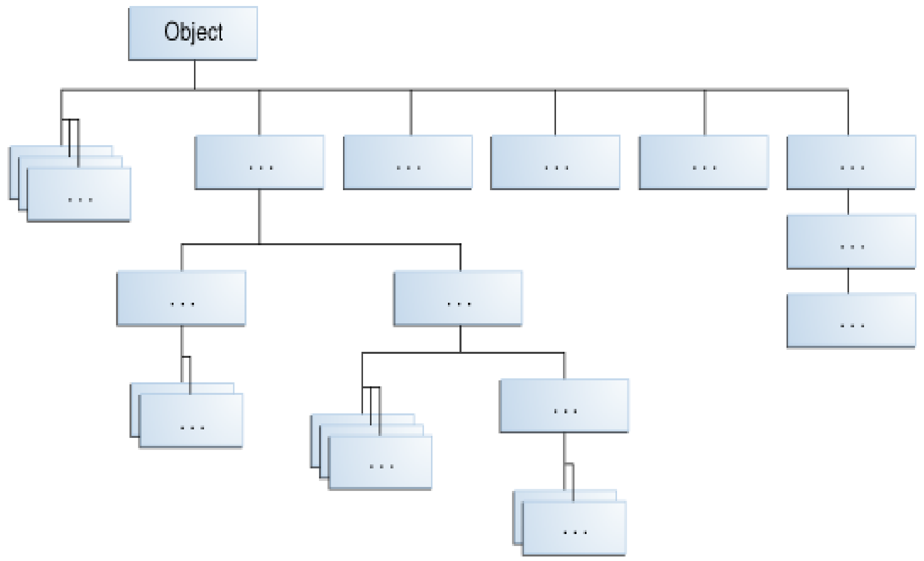
```
class Rectangle extends Figure {
                    ??
          }

class Triangle  extends Figure {
                    ? ?
          }
```

Object class

Introduction

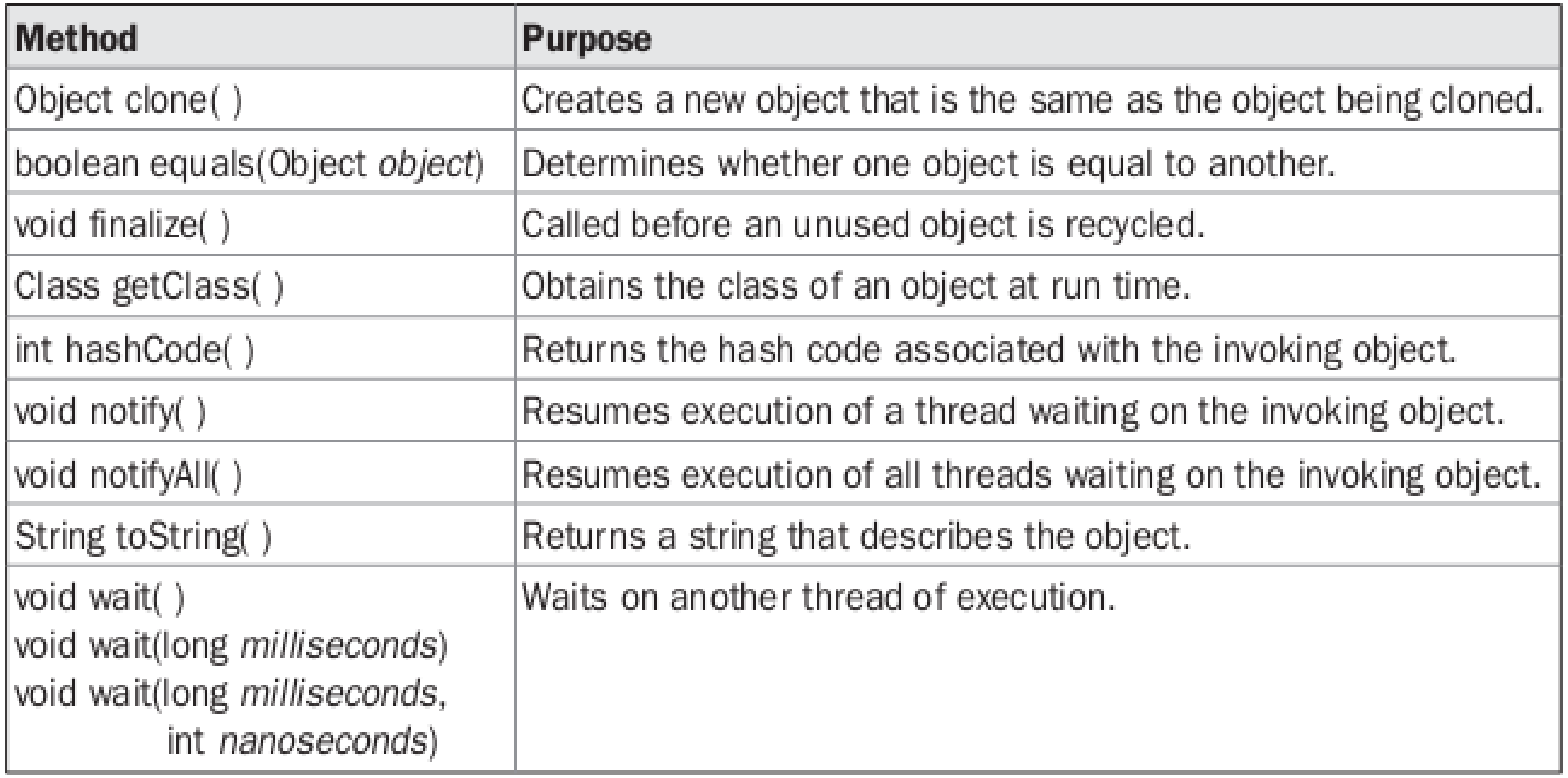
- Object class defined by Java is a super class of all other classes, in the absence of any other explicit superclass
- A reference variable of type Object can refer to an object of any class
- This is defined in the **java.lang** package



Object Oriented Analysis and Design using Java

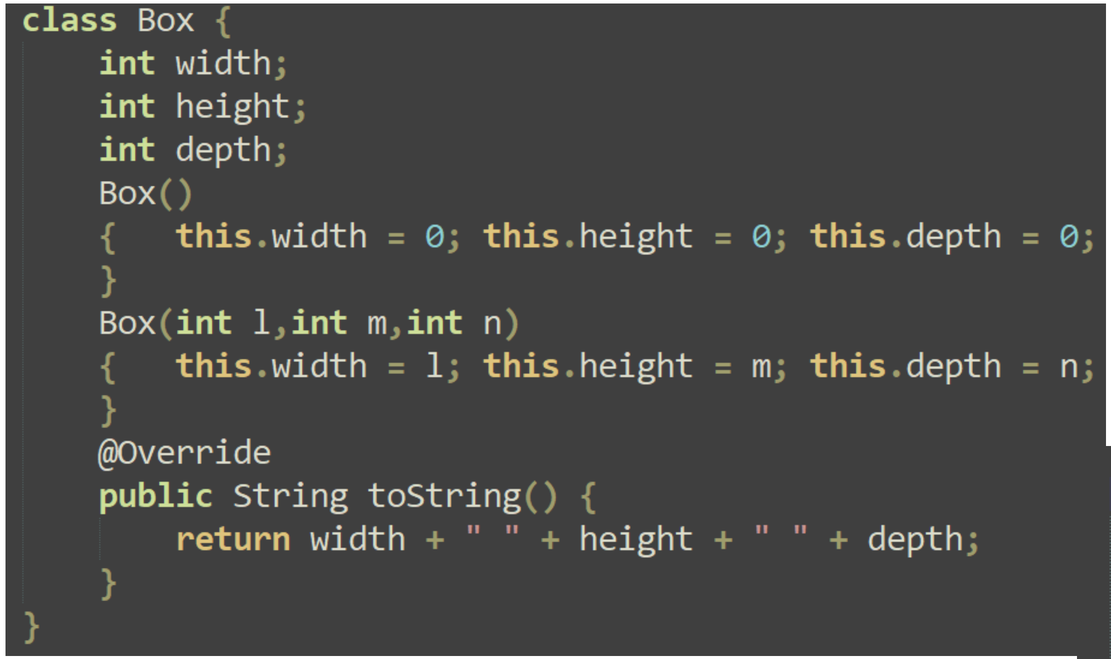
Object class

Methods: Object class defines some methods, which are available in every object

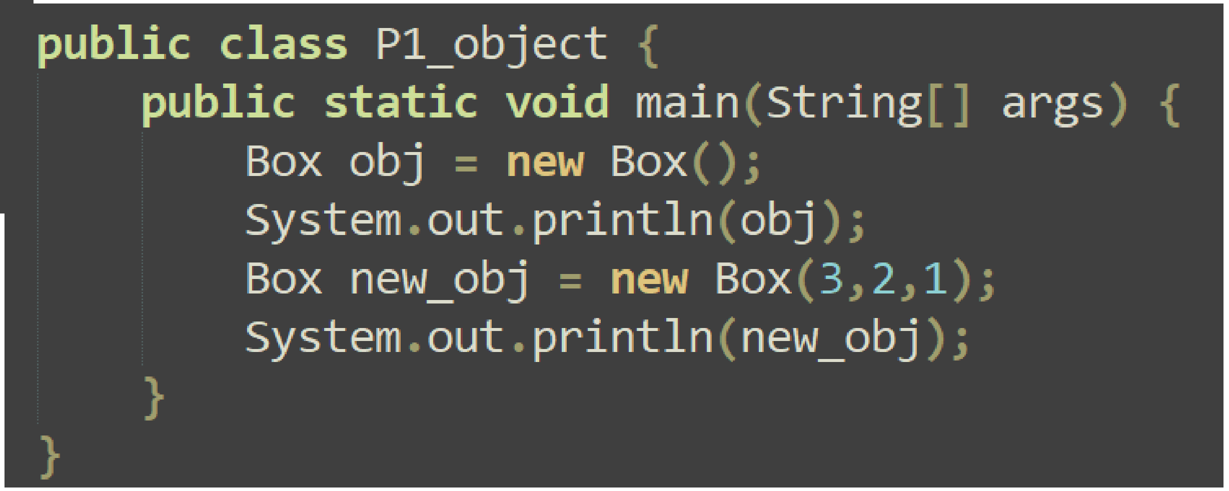


Method	Purpose
Object clone( )	Creates a new object that is the same as the object being cloned.
boolean equals(Object *object*)	Determines whether one object is equal to another.
void finalize( )	Called before an unused object is recycled.
Class getClass( )	Obtains the class of an object at run time.
int hashCode( )	Returns the hash code associated with the invoking object.
void notify( )	Resumes execution of a thread waiting on the invoking object.
void notifyAll( )	Resumes execution of all threads waiting on the invoking object.
String toString( )	Returns a string that describes the object.
void wait( ) void wait(long *milliseconds*) void wait(long *milliseconds*, int *nanoseconds*)	Waits on another thread of execution.

Coding example – 1: Demo of overriding toString() function

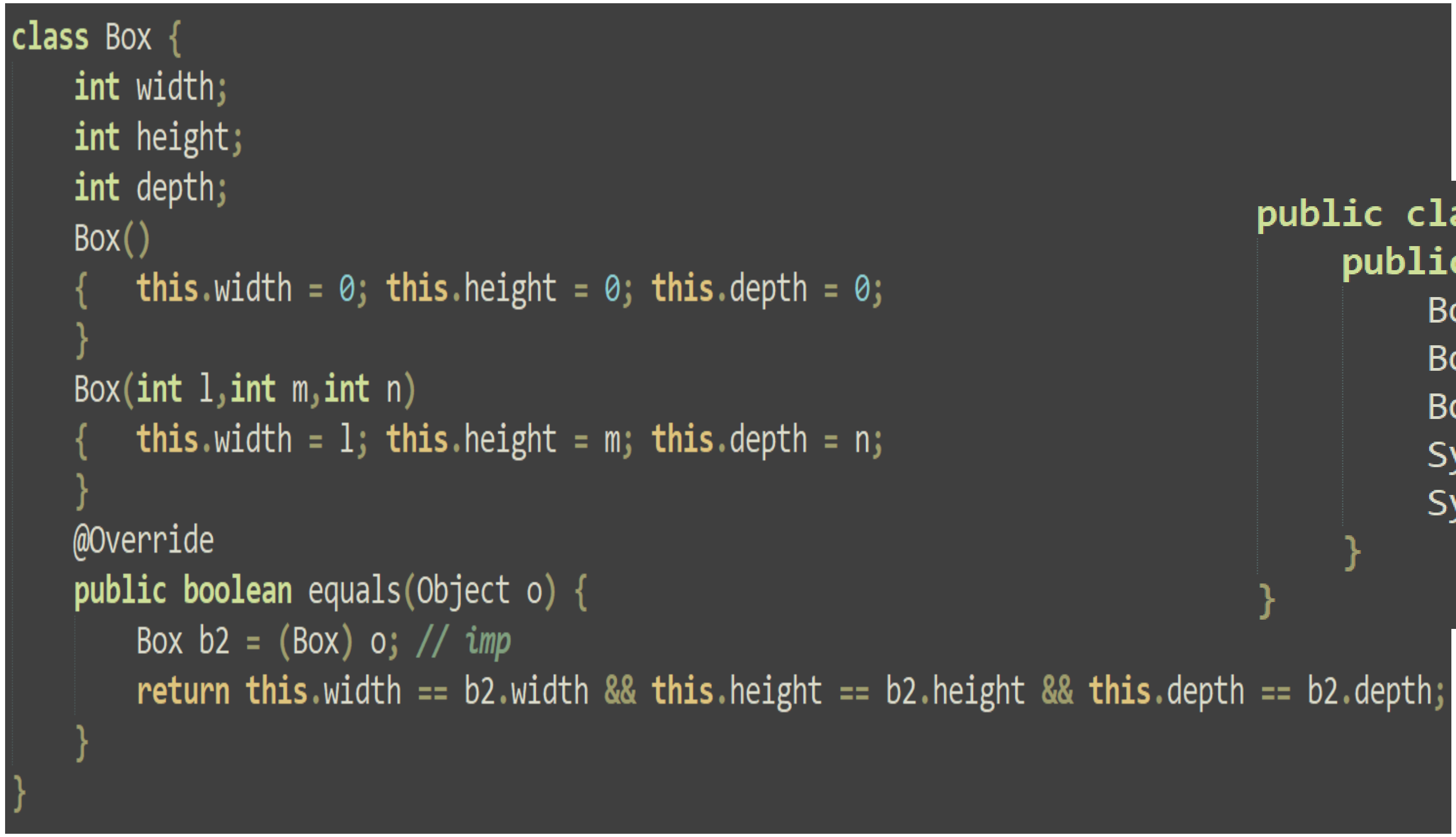


```java
class Box {
    int width;
    int height;
    int depth;
    Box()
    {   this.width = 0; this.height = 0; this.depth = 0;
    }
    Box(int l,int m,int n)
    {   this.width = l; this.height = m; this.depth = n;
    }
    @Override
    public String toString() {
        return width + " " + height + " " + depth;
    }
}
```

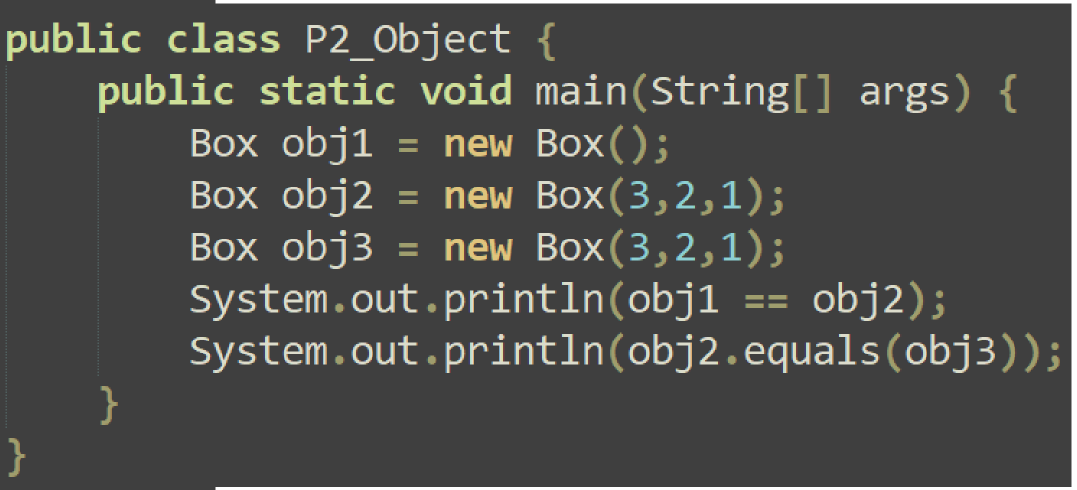


```java
public class P1_object {
    public static void main(String[] args) {
        Box obj = new Box();
        System.out.println(obj);
        Box new_obj = new Box(3,2,1);
        System.out.println(new_obj);
    }
}
```

Coding example – 2: Demo of overriding of equals() function



```java
class Box {
    int width;
    int height;
    int depth;
    Box()
    {   this.width = 0; this.height = 0; this.depth = 0;
    }
    Box(int l,int m,int n)
    {   this.width = l; this.height = m; this.depth = n;
    }
    @Override
    public boolean equals(Object o) {
        Box b2 = (Box) o; // imp
        return this.width == b2.width && this.height == b2.height && this.depth == b2.depth;
    }
}
```



```java
public class P2_Object {
    public static void main(String[] args) {
        Box obj1 = new Box();
        Box obj2 = new Box(3,2,1);
        Box obj3 = new Box(3,2,1);
        System.out.println(obj1 == obj2);
        System.out.println(obj2.equals(obj3));
    }
}
```

THANK YOU

Prof. Sindhu R Pai
Department of Computer Science and Engineering

sindhurpai@pes.edu

+91 8277606459