Object as a Superclass

- **`Object`** is the root of the class hierarchy
  - Every *class* has **`Object`** as a superclass
- All classes inherit the methods of Object
  - But may override them

TABLE 3.2
Methods of Class `java.lang.Object`

Method	Behavior
`Object clone()`	Makes a copy of an object.
`boolean equals(Object obj)`	Compares this object to its argument.
`int hashCode()`	Returns an integer hash code value for this object.
`String toString()`	Returns a string that textually represents the object.

The **`toString()` Method**

- The Object's toString() method returns a String representation of the object, which is very useful for debugging.

- You should always override **`toString`** method if you want to print object state

- If you do *not* override it:
  - **`Object.toString`** will return a **`String`**
  - Just not the **`String`** you want!

    Example: **`ArrayBasedPD@ef08879`**

    ... The name of the class, `@`, instance's hash code

The `equals()` Method

- Compares two objects for equality and returns true if they are equal.
- The equals() method provided by Object tests whether the object references are equal—that is, if the objects compared are the exact same object.
- To test whether two objects are equal in the sense of containing the same information, you must override the equals() method.

The `getClass()` Method

- The getClass() method returns a Class object, which has methods you can use to get information about the class, such as its name (getSimpleName()), its superclass (getSuperclass()), etc..

The `hashCode()` Method

- The value returned by hashCode() is the object's hash code, which is the object's memory address in hexadecimal.
- By definition, if two objects are equal, their hash code must also be equal.
- If you override the equals() method, you must also override the hashCode() method as well.

Casting Objects

- *Casting* obtains a reference of different, but *matching,* type
- Upcasting
  - Casting from a subclass to a superclass is called upcasting

```
Box box = new Box();
Object obj = box;
```

- Upcasting is implicitly performed by the compiler.
- Casting *does not change* the object!

Operations Determined by Type of Reference Variable

- Variable can refer to object whose type is a *subclass* of the variable's declared type
- Type of the *variable* determines what operations are legal
- Java is *strongly typed*
  - Compiler always verifies that variable's type includes the class of every expression assigned to the variable

```
Object obj= new Box(5,5,);
obj.area();                    //  compile-time error.
```

Casting Objects

- *Downcasting*
  - Casting from a superclass to a subclass is called downcasting
  - Downcating should be done explicitly

```
Object obj= new Box(); //upcasting
Box box= (Box)obj;     //downcasting
```

- Checks *at run time* to make sure it's ok
- If not ok, throws **`ClassCastException`**

Casting Objects

- **ClassCastException**
  - **may occur during downcasting**

```
Object obj= "Hello"; //upcasting
Box box= (Box)obj;    //downcasting
```

- Since String is not a Box, an error will occur during runtime

`instanceof` operator

- **`instanceof`** can guard against **`ClassCastException`**

```
Object obj = ...;
if (obj instanceof Box) {
  Box box = (Box)obj;
  int area= box.area();
  ...;
} else {
  ...
}
```

Abstract Methods and Classes

- An abstract class is a class that is declared abstract
  - it may or may not include abstract methods.

- An abstract method is a method that is declared without an implementation (without braces, and followed by a semicolon), like this:

```
abstract void moveTo(double deltaX, double deltaY);
```

Abstract Methods and Classes

```
public abstract class Shape{

    // declare fields

    // declare nonabstract methods

    abstract void calculateArea();
    abstract void calculatePerimeter();
}
```

Abstract Methods and Classes

- When an abstract class is subclassed,
  - the subclass usually provides implementations for all of the abstract methods in its parent class.
  - if it does not, then the subclass must also be declared abstract.

Abstract Methods and Classes

```
class Circle extends Shape{
    void calculateArea() {
        ...
    }
    void calculatePerimeter() {
        ...
    }
}
```

What You Can Do in a Subclass

- The inherited fields can be used directly, just like any other fields.
- You can declare a field in the subclass with the same name as the one in the superclass, thus hiding it (not recommended).
- You can declare new fields in the subclass that are not in the superclass.

What You Can Do in a Subclass

- The inherited methods can be used directly as they are.
- You can write a new instance method in the subclass that has the same signature as the one in the superclass, thus overriding it.
- You can write a new static method in the subclass that has the same signature as the one in the superclass, thus hiding it.

What You Can Do in a Subclass

- You can declare new methods in the subclass that are not in the superclass.
- You can write a subclass constructor that invokes the constructor of the superclass, either implicitly or by using the keyword super

Other Issues

- Except for the Object class, a class has exactly one direct superclass.

- The Object class is the top of the class hierarchy. All classes are descendants from this class and inherit methods from it. Useful methods inherited from Object include
  - toString(), equals(), clone(), and getClass().

Other Issues

- You can prevent a class from being subclassed by using the final keyword in the class's declaration.
- Similarly, you can prevent a method from being overridden by subclasses by declaring it as a final method.
- An abstract class can only be subclassed; it cannot be instantiated. An abstract class can contain abstract methods

References

- http://math.hws.edu/javanotes/
- http://ocw.mit.edu/courses/electrical-engineering-and-computer-science/6-092-introduction-to-programming-in-java-january-iap-2010/lecture-notes/
- https://www.tiobe.com/tiobe-index/
- Head First Java 2nd Edition
- Java a Beginner's guide 7th Edition