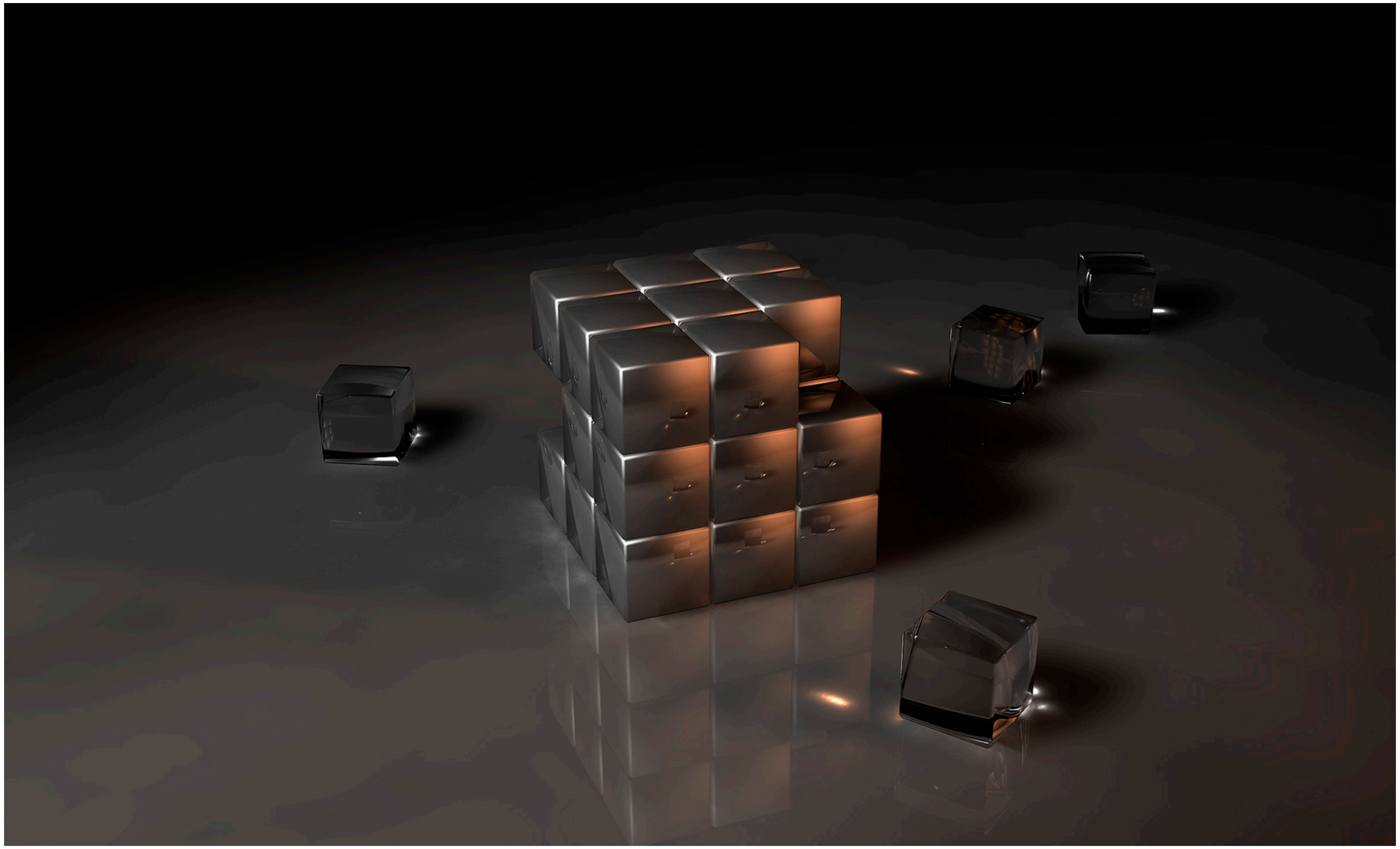


Deep into Java Oop

2019 Lecture 1

Object Oriented Programming

- Java object is mapping the the O in the OOP
- Almost everything in Java are objects
- Object and classes gives a way to describe the programming objects
- It contains instance variables, method and programming logics

Everything we missed in oop

- Detailed in static
- final keyword
- Detailed in constructor
- Detailed into public private
- How to “view” a class
- THE Java Original Object
- Override and Overload
- Class reflection
- Class compare
- Introduction of Concept of inheritance

```
public class SampleClass {

    // can be only access with initialize a variable.
    public int intAccess = 10;

    public SampleClass() {
        // constructor
    }

    public void Method() {
        // can be access with initialize a variable
    }
}
```

```
public class SampleCaller {

    public static void main(String[] args) {
        SampleClass obj = new SampleClass();

        int intValue = obj.intAccess;
        obj.Method();
    }
}
```

```
public class SampleClass {

    // can be access from outside with ClassName.
    public static int intClassLevelAccess = 10;

    public SampleClass() {
        // constructor
    }

    public static void classLevelMethod() {
        // can be access with class name
    }
}
```

```java
public class SampleCaller {

    public static void main(String[] args) {
        SampleClass obj = new SampleClass();
        // there are nothing that we defined that can be accessed here


        int intValue = SampleClass.intClassLevelAccess;
        SampleClass.classLevelMethod();
    }
}
```

Final KeyWord

- describe something that cannot be changed at all
- Can describe a instance variable
- Can describe a method (will cover in latter lecture)
- Can describe a class (will cover in latter lecture)
- Usage:
  - If a public variable has to be exposed, make it final to protected
  - A final variable cannot be modified upon initialed assign
  - A final static variable has to be initialed right away

```java
public class SampleClass {

    // can be access from outside with ClassName.
    public final int finalVariable;
    public final static int sharedFinalVariable = 20;

    public SampleClass() {
        // constructor
        finalVariable = 10;
    }

    public static void classLevelMethod() {
        // can be access with class name
    }
}
```

```
public class SampleCaller {

    public static void main(String[] args) {
        SampleClass obj = new SampleClass();
        obj.finalVariable = 20; // wrong


        int intValue = SampleClass.sharedfinalVariable;
        SampleClass.sharedfinalVariable = 100; // wrong
        SampleClass.classLevelMethod();
    }
}
```

Almost everything in Java are objects

```
public class SampleCaller {




    public static void main(String[] args) {
        // Why would this work.
    }
}
```

```
public class SampleCaller {

    public SampleCaller() {
         // every class has a default constructor
        // default constructor has no parameters
        // default constructor has no implementation body
    }

    public static void main(String[] args) {

    }
}
```

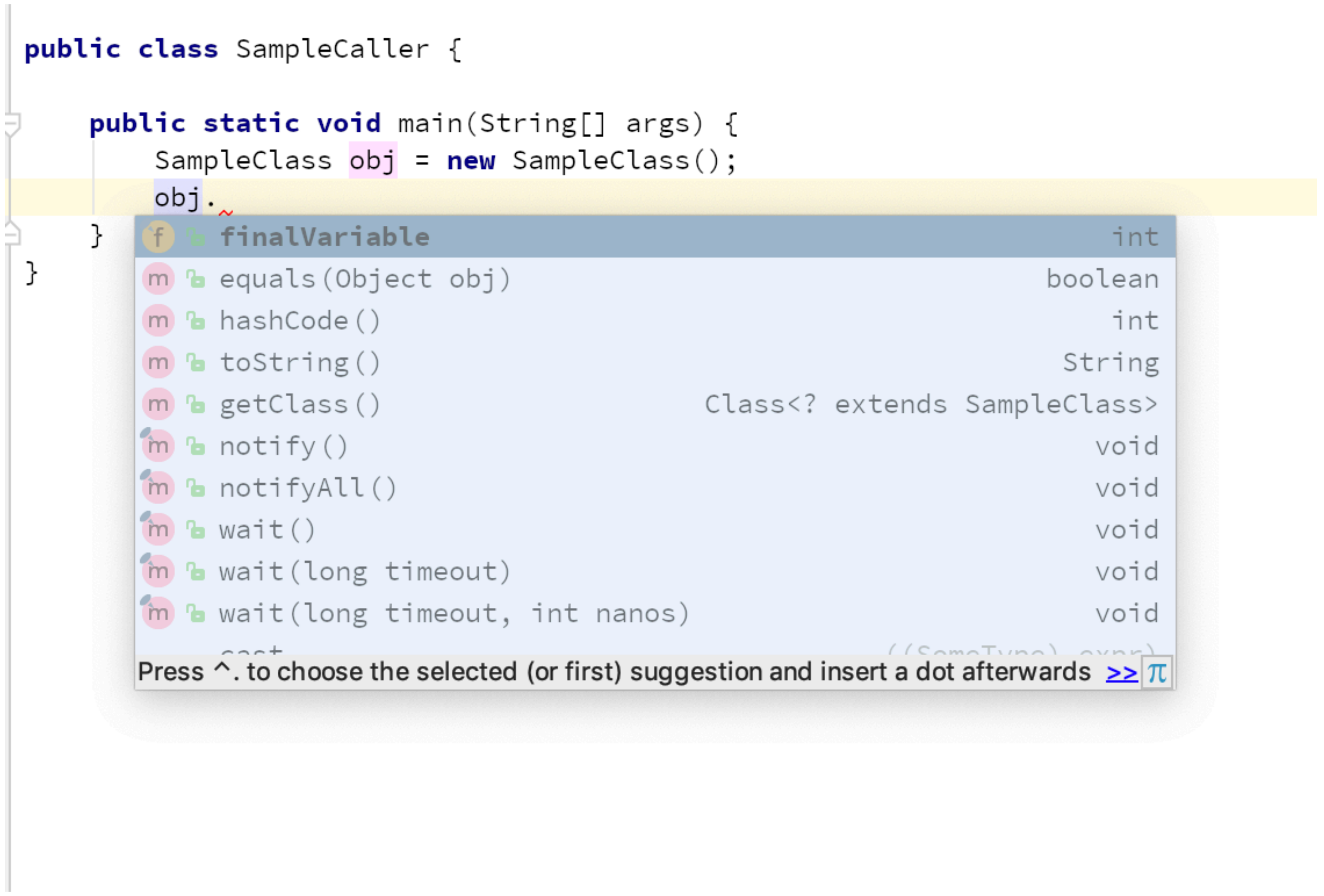
Constructor

- Constructor is the bridge to create the object from the the class
- Every class has a default constructor that with no any parameter and implementation body

Public && Private

- Public
  - Gives access for everything it declares
  - Can describe instance variable
  - Can describe method
  - Can describe class
- Private
  - Limit everything only private access only for the class
  - Can describe instance variable
  - Can describe method
  - Can describe class (will cover in the later class)

Almost everything in Java are objects
-
The original Java Object



```
public class SampleCaller {

    public static void main(String[] args) {
        SampleClass obj = new SampleClass();
        obj.
    }
}
```

```
finalVariable                                    int
equals(Object obj)                               boolean
hashCode()                                       int
toString()                                       String
getClass()                  Class<? extends SampleClass>
notify()                                         void
notifyAll()                                      void
wait()                                           void
wait(long timeout)                               void
wait(long timeout, int nanos)                    void
```

Press ^. to choose the selected (or first) suggestion and insert a dot afterwards >> π

The Object

- The parents class of all java classes
- Object.java
- Contains basic default method for objects
- All java classes extends Object class, but don’t have to write this down

```java
public class Object {

    public native int hashCode();

    public String toString() {
        return getClass().getName() + "@" + Integer.toHexString(hashCode());
    }

    public boolean equals(Object obj) {
        return (this == obj);
    }
}
```

```
public class SampleClass extends Object {

    // can be access from outside with ClassName.
    public final int finalVariable;
    public final static int sharedfinalVariable = 20;

    public SampleClass() {
        // constructor
        finalVariable = 10;
    }

    public static void classLevelMethod() {
        // can be access with class name
    }
}
```

Override

- A way that child class take over the default behaviour of parents class
- Use @Override to declare it. Can also ignore this

```java
public class SampleClass extends Object {

    // can be access from outside with ClassName.
    public String content;
    public int intValue;

    public SampleClass(String inputcontent, int inputValue) {
        // constructor
        content = inputcontent;
        intValue = inputValue;
    }

    public static void main(String[] args) {
        SampleClass obj = new SampleClass();

        System.out.print(obj);
    }
}
```

```
Midterm.SampleClass@61bbe9ba
```

```java
public class SampleClass extends Object {

	// can be access from outside with ClassName.
	public String content;
	public int intValue;

	public SampleClass(String inputcontent, int inputValue) {
		// constructor
		content = inputcontent;
		intValue = inputValue;
	}

	@Override
	public String toString() {
		return "Print: " + content + " " + intValue;
	}

	public static void main(String[] args) {
		SampleClass obj = new SampleClass("Test", 10);

		System.out.print(obj);
	}
}
```

```
Print: Test 10
```

Overload

- A way to provided different style of function with same method name within the same class
- Has to be the same name
- With same return type
- Only difference allowed is parameter

```java
public class SampleClass extends Object {

	// can be access from outside with ClassName.
	public String content;
	public int intValue;

	public SampleClass(String inputcontent, int inputValue) {
		// constructor
		content = inputcontent;
		intValue = inputValue;
	}

	public int calculate(int a) {
		return a++;
	}

	public int calculate(int a, int b) {
		return a + b;
	}

//  this is not allowed
//	public boolean calculate(int a, int b) {
//		return a + b;
//	}

//  this is not allowed
//	public int calculate(int a, int b) {
//		return a - b;
//	}

}
```

Constructor Overload

- Constructor is a special types of a method, so overload also applies
- Constructor overload is more common than normal functions
- Provides different ways to initial the object

```
public class SampleClass extends Object {

    // can be access from outside with ClassName.
    public String content;
    public int intValue;

    public SampleClass() {

    }

    public SampleClass(String inputcontent) {
        content = inputcontent;
    }

    public SampleClass(int inputValue) {
        intValue = inputValue;
    }

    public SampleClass(String inputcontent, int inputValue) {
        content = inputcontent;
        intValue = inputValue;
    }
}
```

Class Reflection: this

- Keyword
- Have access to everything of the current class
- Represent the current class

```
public class SampleClass extends Object {

    // can be access from outside with ClassName.
    public String content;
    public int intValue;

//  wrong way to initial
//  public SampleClass(String content, int intValue) {
//      content = content;
//      intValue = intValue;
//  }

    public SampleClass(String content, int intValue) {
        this.content = content;
        this.intValue = intValue;
    }

    @Override
    public String toString() {
        return this.content;
    }
}
```

Compare between objects

- Objects compare are very different
- Object cannot directly use == to compare
  - == compares the
- Use the equal method to compare object

```
public class Object {

    public native int hashCode();

    public String toString() {
        return getClass().getName() + "@" + Integer.toHexString(hashCode());
    }

    public boolean equals(Object obj) {
        return (this == obj);
    }
}
```

```
public class SampleClass extends Object {

    public String content;
    public int intValue;

    public SampleClass(String content, int intValue) {
        this.content = content;
        this.intValue = intValue;
    }


    public static void main(String[] args) {
        SampleClass obj1 = new SampleClass("Test", 10);
        SampleClass obj2 = new SampleClass("Test", 10);

        System.out.println(obj1 == obj2);
        System.out.println(obj1.equals(obj2));
    }
}
```

```
false
false
```

```
public class SampleClass extends Object {

    public String content;
    public int intValue;

    public SampleClass(String content, int intValue) {
        this.content = content;
        this.intValue = intValue;
    }


    public static void main(String[] args) {
        SampleClass obj1 = new SampleClass("Test", 10);
        SampleClass obj2 = new SampleClass("Test", 10);

        System.out.println(obj1 == obj2);
        System.out.println(obj1.equals(obj2));
    }
}
```

```
false
false
```

```java
public class SampleClass extends Object {

    public String content;
    public int intValue;

    public SampleClass(String content, int intValue) {
        this.content = content;
        this.intValue = intValue;
    }

    @Override
    public boolean equals(Object obj) {
        if (((SampleClass)obj).intValue == this.intValue) {
            return true;
        } else {
            return false;
        }
    }

    public static void main(String[] args) {
        SampleClass obj1 = new SampleClass("Test", 10);
        SampleClass obj2 = new SampleClass("Test", 10);

        System.out.println(obj1 == obj2);
        System.out.println(obj1.equals(obj2));
    }
}
```

```
false
true
```

Preview on inheritance

- The way a class inherit the allowed behaviour and allowed attribute of the parent class
- Use key word extends
- public / private / protected controls rules of inheritance