


Chapter 2: Java OOP I

Yang Wang
wyang AT njnet.edu.cn

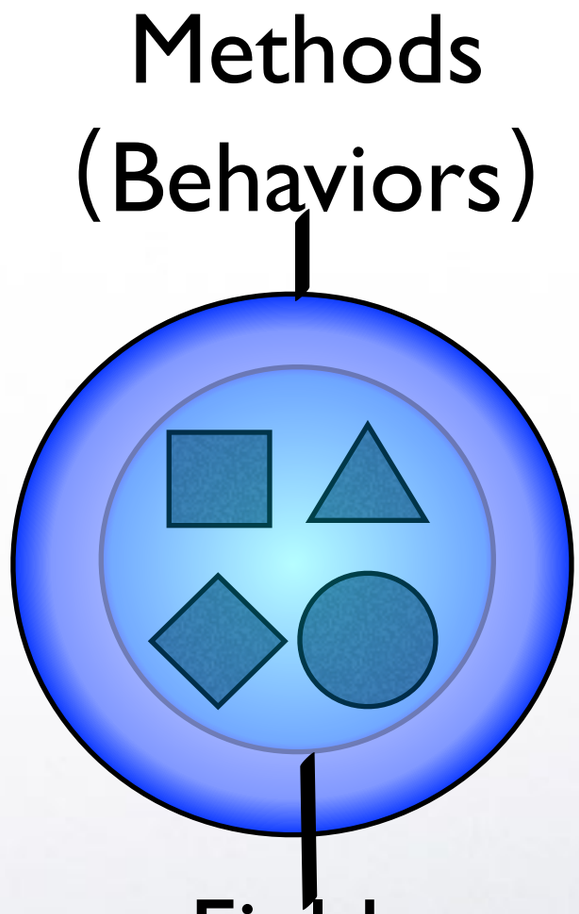
Outline

- OO Concepts
- Class and Objects
  - Package
  - Field
  - Method
  - Construct and Initialization
  - Access Control

OO Concepts

Object Oriented

- Object
  - An object is a software bundle of related state and behavior.
  - Software objects are often used to model the real-world objects
    - Bicycle, Human Being,



Object Oriented

- Why Object
  - Modularity (program divided in different objects)
  - Information Hiding (implementation is hided)
  - Code Re-use ??
  - Pluggability and debugging ease (refer to rule 1 and 2)

Object Oriented

- Class
  - A class is a blueprint or prototype from which objects are created.
    - many individual objects are all of the same kind
    - encapsulation of behaviors and state
    - describes a hierarchical concepts,
      - human->primate->mammal->animal
    - IN Java, the hierarchy of classes is a tree

Object Oriented

- OO Techniques:
  - Abstraction
  - Inheritance
  - Polymorphism

Object Oriented

```
public class Bird{
      public void tweet(){System.out.println("JiuJiu~");}
      public void fly(Place A, Place B){}
}
```

```
public class Parrot extends Bird{
      public void tweet(){System.out.println("Hello~");}
}

......

Bird p = new Parrot();
p.fly(JLH, SPL);
p.tweet();
```

Class and Objects

A Simple Class

```
package cn.edu.seu.cose.javacourse.ch02;
public class Person{
    private String name;
    private int age;
    public Person(String name, int age){
        this.name = name;
        this.age = age;
    }
    public void greet(){
        System.out.println("Hello, I am" + name)
            + " , and I am " + age + " years old.";
    }
    public static void main(String[] args){
        Person tom = new Person("Tom", 18);
        tom.greet();
    }
}
```

Class and Objects

- Package name/ Class name
- import
- Members
  - Field - static / non-static
  - Method - static / non-static
- Access Modifier(Class / Field / Method)
  - abstract / final
  - public / protected / private /default

Class and Object - Package

- Package is a set of Classes
  - To avoid classes with same names
  - To manage classes
    - in your project vs. in internet
- Define a package
  - lowercase
  - project / subproject
    - package javacourse;
  - organization domain(reverse) + project + subproject
    - package cn.edu.seu.cose.javacourse;

Class and Object - Package

- how to use a package
  - referring the class in package with full qualified name
    - import java.io.*;
    - import java.io.File;
- JDK Package
  - java.*; javax.*
  - java.lang.* : default imported

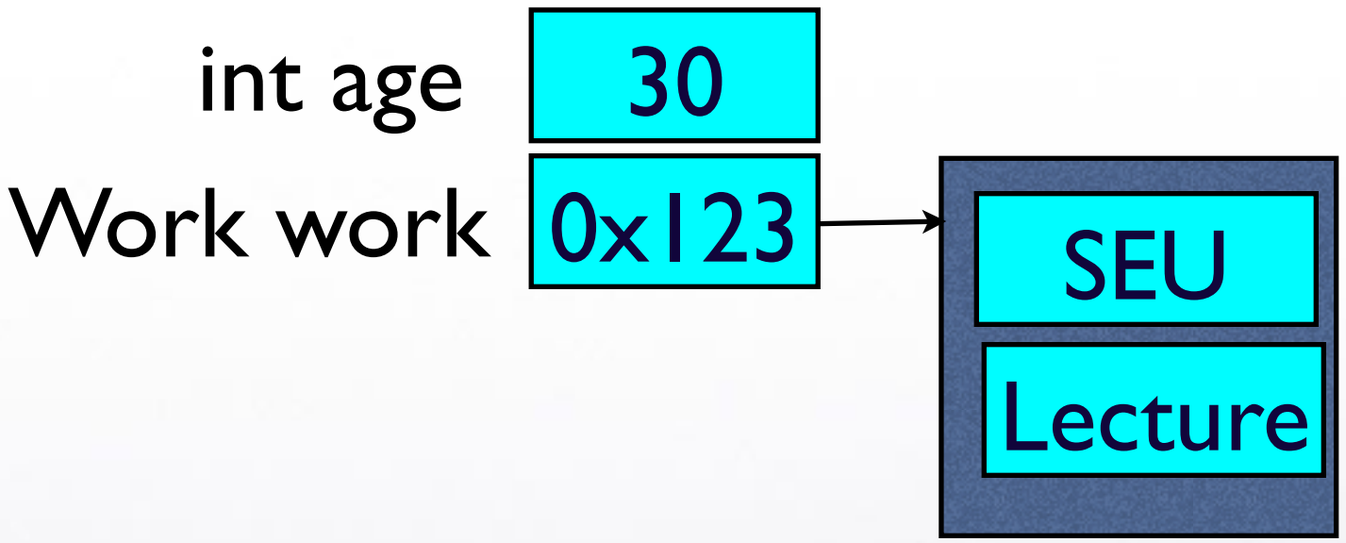
Class and Object – Field

- Define a Field
  - Access Modifier
  - Static Modifier (Optional)
  - Type
  - Name
    - first word lowercase

```
public String name;
private int age;
private Work work;
```

Class and Object – Field

- Type
  - Primary Type
  - Reference



Class and Object – Field

- Declare a reference of an object, but not create it

```
String s;  Person Tom;
```

- Declare a reference of an object, and create the object

```
String s = new String("Hello World");
Person tom = new Person(“Tom”, 18);
```

Class and Object – Field

- Static
  - Class Variables : static field
  - Object Variables : non-static field
- LifeCycle:
  - static field : as long as the program exist
    - class.field
  - non-static field : as long as object exist
    - object.field
  - local variables : in function

```
public static int numOfHuman;
public int age;
```

Class and Object – Method

- Define a Method
  - Access Modifier
  - Static Modifier (Optional)
  - Return Type
  - Name
  - Parameter List (Type + Name)
  - Method Body

```
public class Person{
    public int height;
    // 初始化
    public boolean isHigh(){
        if(height>180)
            return true;
        else
            return false;
    }

    public boolean higherThan(Person someone){
        if(height>someone.height)
            return true;
        else
            return false;
    }
}
```

Class and Object – Method

- Static Method
  - class Method

```
public class Calculator{
	public static int add(int a, int b){
		return a+b;
	}
}
...
System.out.println(Calculator.add(1 + 2));
```

Class and Object – Method

- Static
- Class Methods : static methods
- Object Methods : non-static methods
- LifeCycle:
  - static field : as long as the program exist
    - class.method()
  - non-static field : after object exist
    - object.method()

Class and Object – Method

- public static void main(String args[])
  - why static : Lifecycle

Class and Object – Overloading

- Method Overloading(重载)
  - Method Name
  - Method Signature
    - Method name
    - Number of Parameters
    - Types of Parameters
- Multiple methods with same name in a class: OK (Overloading)
- Multiple methods with same signature in a class: No!
- Signature does not include return type, because signature reflects the specification of behavior, not the result of behavior.

Class and Object – Overloading

- Examples:

```
public void test(int a, String b){}

public String test(int i, String j){}

public String test(String r, int s){}

public void test(int x, String y, int c){}
```

Class and Object – Overloading

- Examples:

```
public void test(int a, String b){}
```

```
public void test(String a, int b){}      Right, Overloading Method

public String test(int i, String j){}    Wrong, Duplicate Method

public String test(String r, int s){}    Right, Overloading Method

public void test(int x, String y, int c){}  Right, Overloading Method
```

Class and Object – Parameters

- Forget them:
  - Formal Parameter?
  - Actual Parameter?
  - Pass by Value?
  - Pass by Reference?
- In Java, the Copy of Parameter is passed.
- What is copied?
  - For primary types, their value is copied
  - For objects, the reference is copied. (What is a reference?)

Class and Object – Parameter

- Try :

```
public class TestClass {
   public static void changeParam(int paramInt){
      paramInt += 5;
   }
   public static void main(String args[]){
      int i = 5;
      TestClass.changeParam(i);
      System.out.println(5);
   }
}
```

i : 5 ?? 10

Class and Object – Parameter

- Try Again:

```
public class Person {
   public String name;
   public Person(String name){
      this.name = name;
   }
}
public class TestClass {
   public static void changeName(Person person){
      person.name = "Jerry";
   }
   public static void main(String args[]){
      Person tom = new Person("Tom");
      changeName(tom);
      System.out.println(tom.name);
   }
}
```

Tom ??

Jerry ??

Class and Object – Parameter

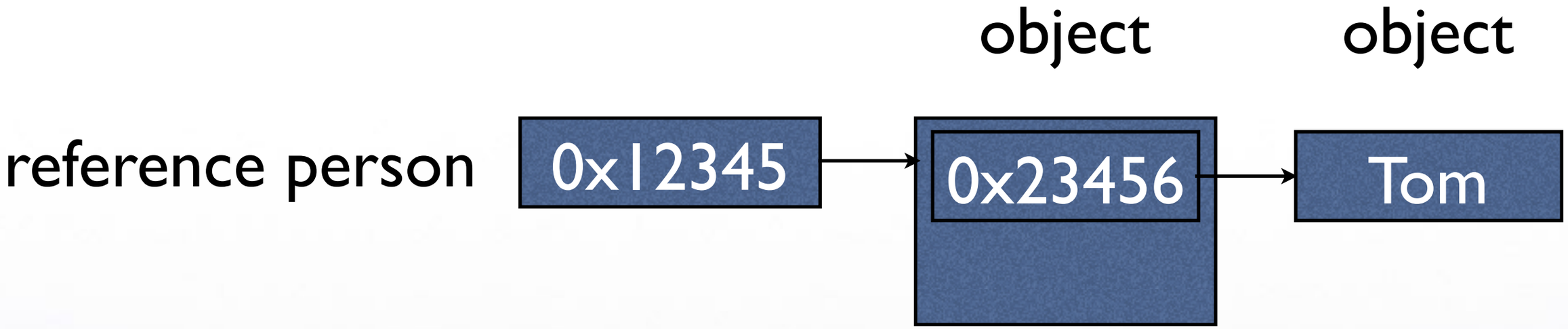
- Try Again:

```
public class Person {
   public String name;
   public Person(String name){
      this.name = name;
   }
}
public class TestClass {
   public static void change(Person person){
      person = null;
   }
   public static void main(String args[]){
      Person tom = new Person("Tom");
      changeName(tom);
      System.out.println(tom.name);
   }
}
```

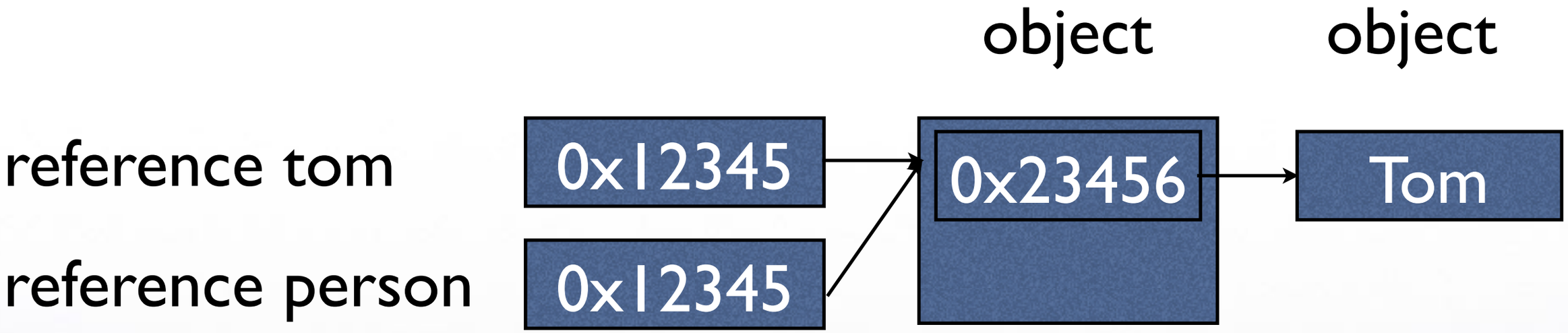
Tom ??

??

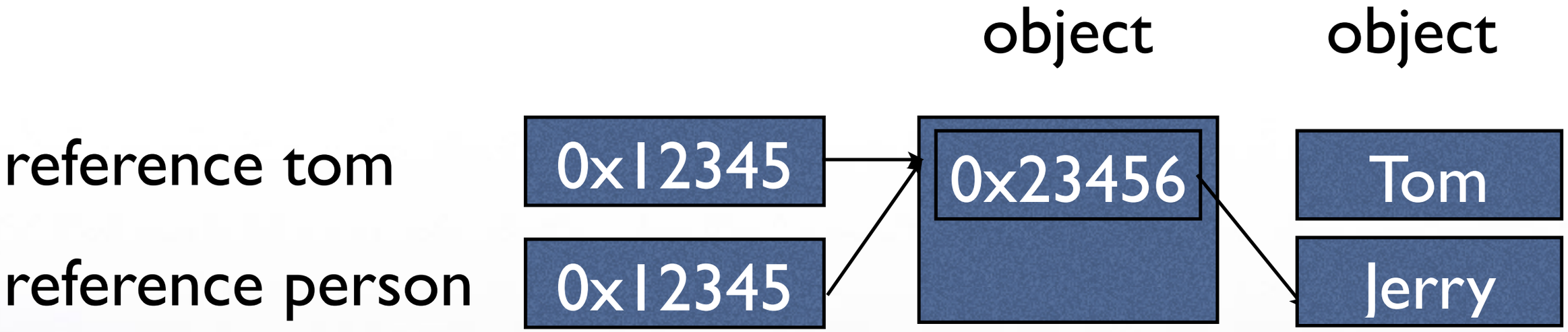
Class and Object – Parameter



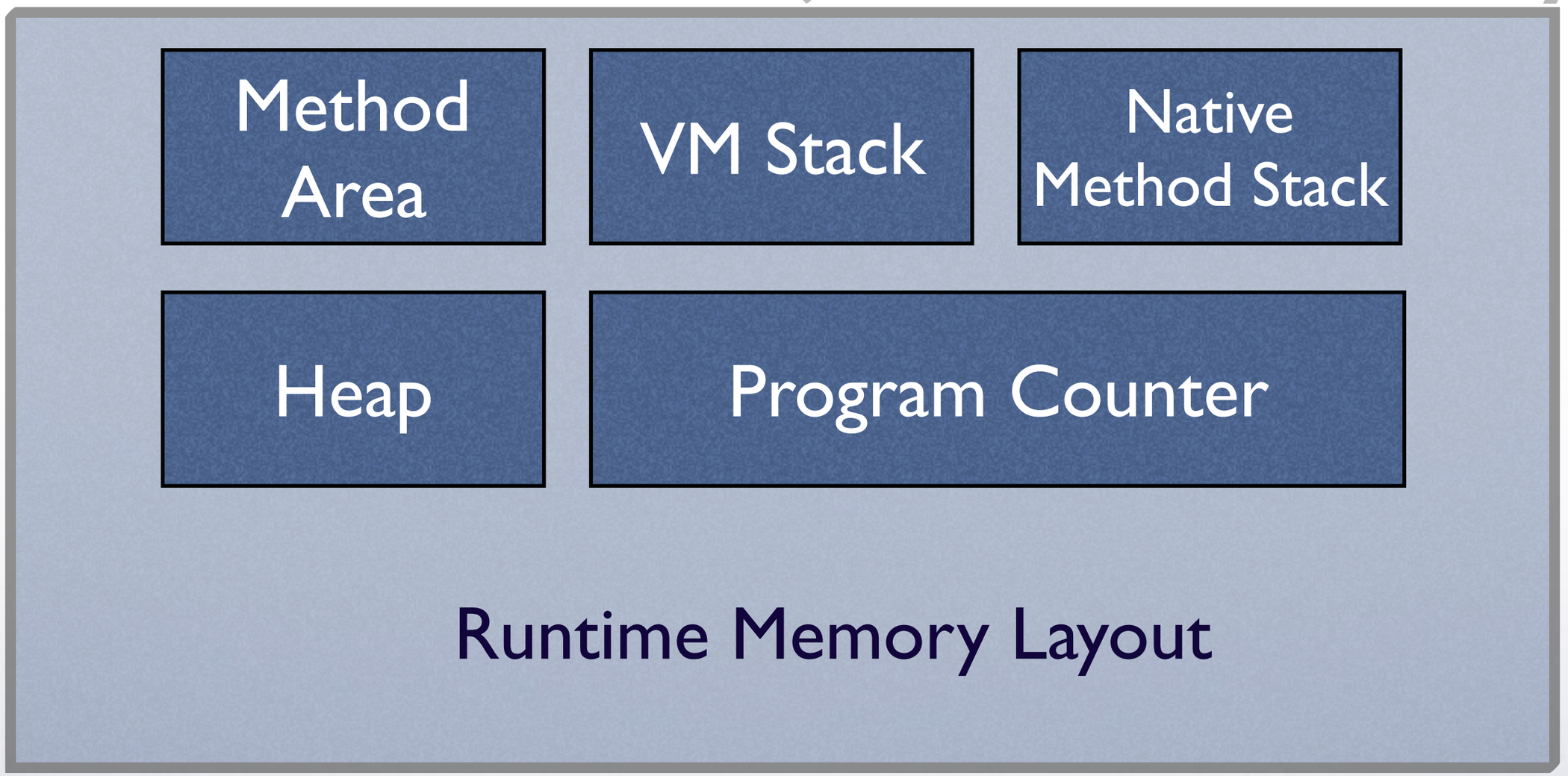
Class and Object – Parameter



Class and Object – Parameter



Class and Object – Memory



Class and Object – Parameter

- Methods with variable number of parameters

```
public class Calculator{
    public static int add(int ...numbers){
        int result = 0;
        for(int i=0; i<numbers.length; i++){
            result += numbers[i];
        }
        return result;
    }
    public static void main(String[] args){
        System.out.println(Calculator.add(10,11));
        System.out.println(Calculator.add(10,11,12));
    }
}
```


Class and Object – Access Control

- Why Do We Need Access Control?
  - Encapsulation
    - Data Hiding
- Without Access Control:
  - Debugging becomes difficult
  - Data and programs become unsafe

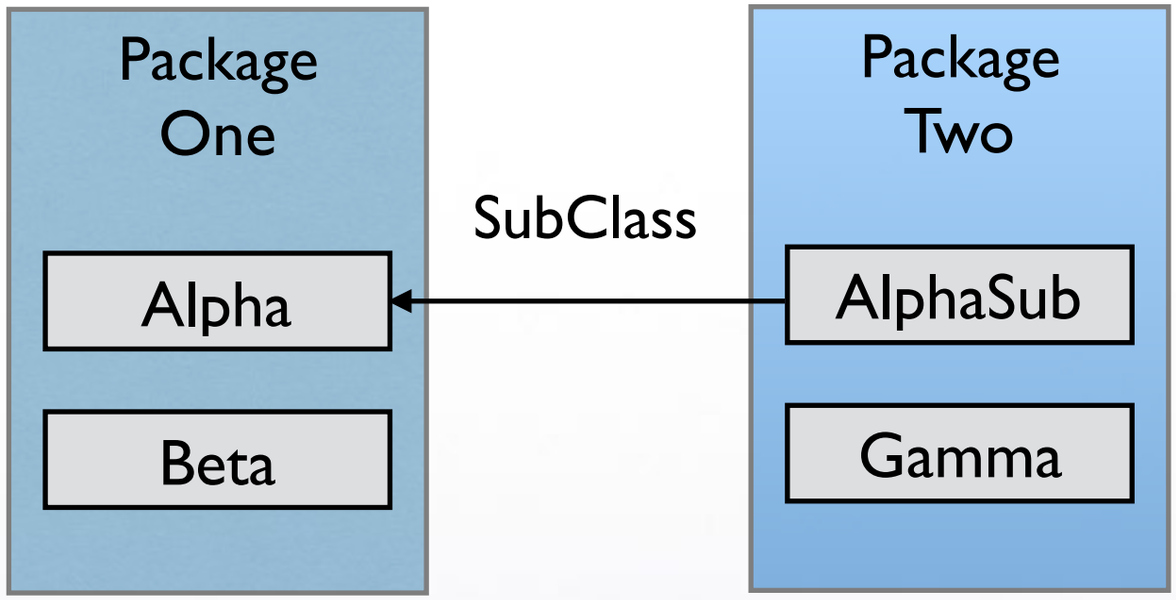
Class and Object – Access Control

- AC Modifier for Classes
  - default (package)
  - public
- AC Modifier for Members
  - default （package）
  - public
  - private
  - protected

Class and Object – Access Control

Modifier	package	world
public	Y	Y
default	Y	N

Class and Object – Access Control



Class and Object – Access Control

Modifer	Alpha	Beta	SubAlpha	Gamma
public				
protected				
default				
private				

Class and Object – Access Control

Modifer	Alpha	Beta	SubAlpha	Gamma
public	Y	Y	Y	Y
protected	Y	Y	Y	N
default	Y	Y	N	N
private	Y	N	N	N

Class and Object – Access Control

Modifer	Class	Package	SubClass	World
public	Y	Y	Y	Y
protected	Y	Y	Y	N
default	Y	Y	N	N
private	Y	N	N	N

Class and Object – Access Control

- Getter and Setter Methods

```
...
private String name;
private int age;
public String getName() {return name;}
public void setName(String name) {this.name = name;}
public int getAge() {return age;}
public void setAge(int age) {
    if(age>150 || age<0){
        age = 0;
        System.out.println("Wrong age!");
    }else{
        this.age = age;
    }
}
...
```


Construction and Initialization

- How to describe the construction of an object in a class?
  - Constructor
    - Default Constructor
    - Constructor with parameters
  - Initialization Block

Construction and Initialization

- Constructor
  - Default
  - With Parameters
  - Constructor Nest
    - this

```
public class Person(){
    public String name;
    public int age;
    public boolean isEducated;
    public Person(){
        this.isEducated = true;
    }
    public Person(String name, int age){
        this();
        this.name = name; this.age = age;
    }
    public Person(String name, int age, boolean isEducated){
        this(name, age);
        this.isEducated = isEducated;
    }
}
```

Construction and Initialization

- Be careful of default Constructor
  - default : no param
  - JVM will give you one for free
    - only when you don’t write any constructor

Construction and Initialization

```
public class JTest2 {

    public static void main(String args[]){
        JTest2 t = new JTest2();
    }

}
```

```
public class JTest2 {
    public JTest2(int x){

    }

    public static void main(String args[]){
        JTest2 t = new JTest2();
    }

}
```

Construction and Initialization

- Initialization Block

```
public class Person{
    public int id;
    public static int counter;
    {
        id = counter++;
    }
    public static void main(String[] args){
        Person tom = new Person();
        Person mike = new Person();
        System.out.println(tom.id);
        System.out.println(mike.id);
    }
}
```

Construction and Initialization

- Static Initialization Block

```
public class Person{
    public int id;
    public static int counter;
    public static int getBeginID(){
        ...  // 从数据库中获取ID
    }
    static{
        counter = getBeginID();
    }
    ...
}
```


Construction and Initialization

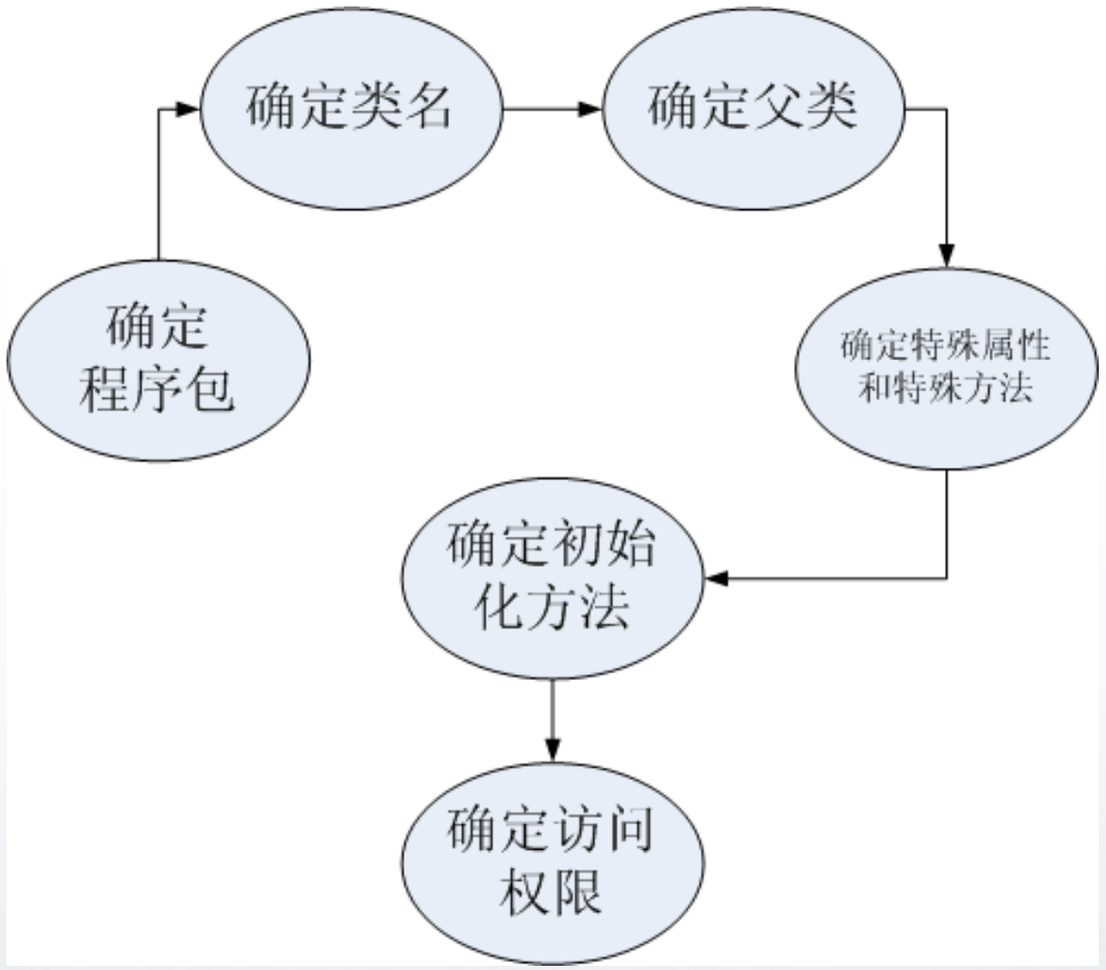
- Destroying Object
  - Java GC (Garbage Collection)
    - Reference Counter
    - finalize()
      - political reason
      - never use it

Class - Object

- All classes in Java inherits java.lang.Object
- All objects in Java have following methods:

```
public boolean equals(Object obj)
public int hashCode()
protected Object clone() throws CloneNotSupportedException
public final Class<?> getClass()
protected void finalize() throws Throwable
public String toString()
```

Last : How to Construct a Class



Forecast

- Abstraction
  - Abstract Class
  - Interface
- Inheritance
- Polymorphism