# Overview

## Java Origins and Ecosystem

Java Card

ME

SE：主要，核心API

MP：服务器

EE：企业服务

跨平台：在Mac上编译的Java代码能够运行在windows上

## Java Language

Class：类

Package：一些外部包

Module

创建类

错误处理

## Java Design

Class

Object

Inheritance

UML:

如何分析和架构应用

## Java APIs

### Collection APIs

List

Sets等等

### Stream APIs

使用lambda表达式，操纵大型数据

### IO APIs

### Concurrency API

多线程

### Persistence API

数据库

### Java in the Enterprise

必须在容器中才能进行部署

# Object-Oriented Concepts

汇编器 assembler

Alan Kay

抽象

对象的基本形式

>> 分析工作，分析必要的属性和方法

封装

将复杂内容进行封装而不用关注内部的实现，确保对象和主程序不会相互影响

>> 仅通过发送消息来实现内部状态的修改

继承

>> 子类重用父类代码的能力

多态

>> 将IF ELSE语句迁移到运行时的虚拟机内

>> 延迟绑定/动态绑定

# What is Java Program

特性：

1. JVM
2. OO Language  
   好处：模块化、实现信息隐藏、代码复用、维护性提高  
     
     
   组成：
3. Java Class：基本的组成部分
4. Java Packages：用于将Class组合起来的东西

IDE：

NetBeans

Read-Evaluate-Print Loop(REPL):

JShell

# Java Text and Numbers

variables

1. initialized with a value
2. change
3. specific type of data

use of variables

1. holding data
2. assigning the value of one variable to another
3. mathematical
4. print

Basic Value type:

int

double

Constants：

**final**

Operators：

1. 括号
2. ++ and --
3. \* and /
4. + and -

Boolean Expressions

Primitives can be compared with relational operators, e.g. >.

Objects can be compared with operations, e.g. equals.

Java Documentation provides the reference for the classes and their operations, e.g. String.

# Java Arrays, conditions and Loop

conditions:

if else

switch

arrays:

initialize

set value

use the args in the main method

loops:

1. while loop
2. do while loop
3. for loop and **enhanced for loop**
4. for each loop

break, continue

# Java Class and Objects

Using UML diagrams：

class name

fields: data types

methods: method parameters

methods:

<access modifier> <return type> <method name> (<parameter list>) {

// body

// return statement

}

Using Method Overloading:

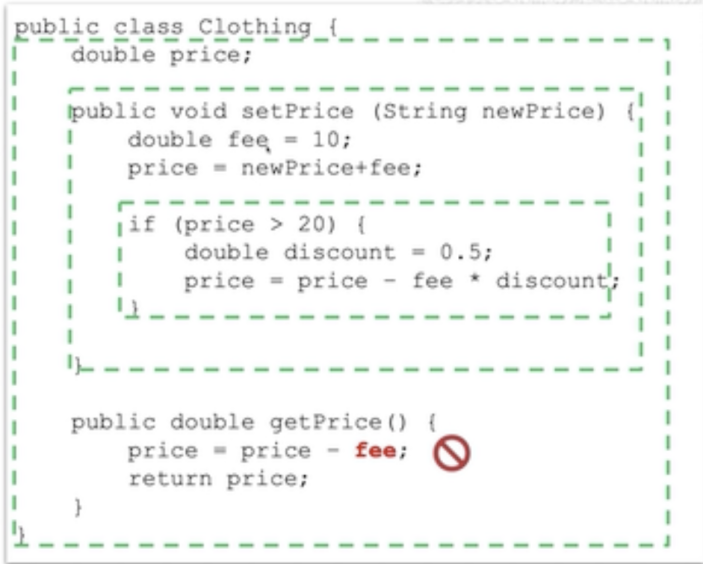
have the same name

have the different parameter list

## Scope and Access

Variable:

1. instance variables
2. local variables
3. “block” variables



variable shadowing:

this: disambiguate from other parameters

access modifiers:

public

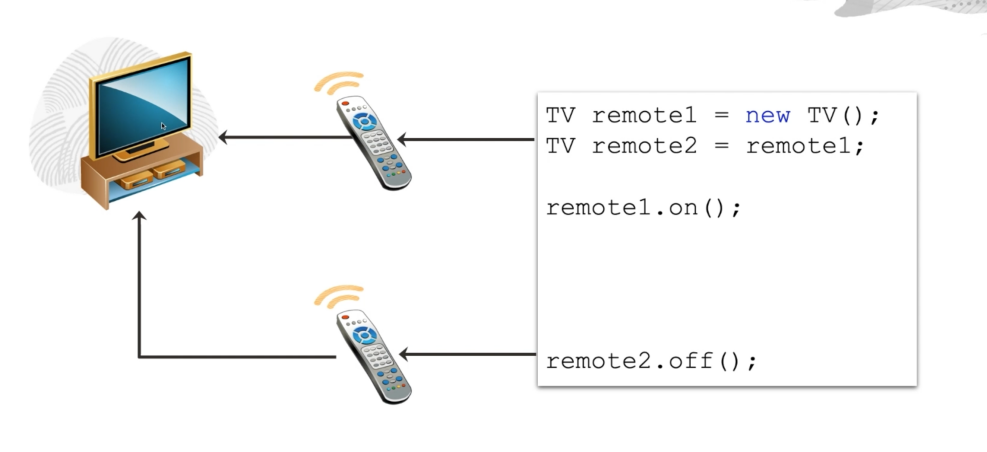
private

Exercise 5-1 Apply Encapsulation

Exercise 5-2 Overload a Method

Exercise 5-3 Associate "Customer" with "Clothing"

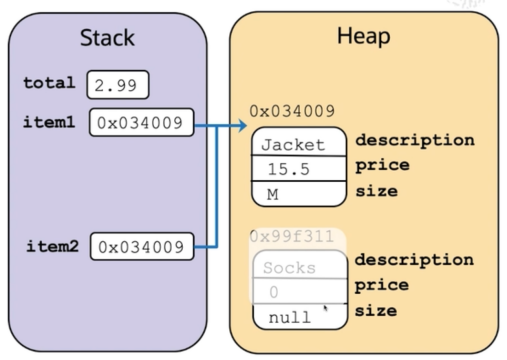
## Objects Reference and Memory Allocation



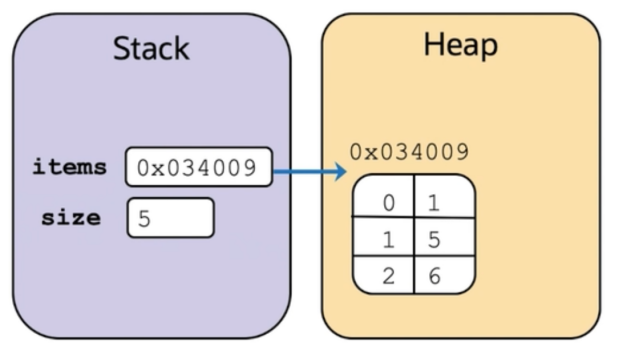
variable in memory: Stack

objects in memory: Heap

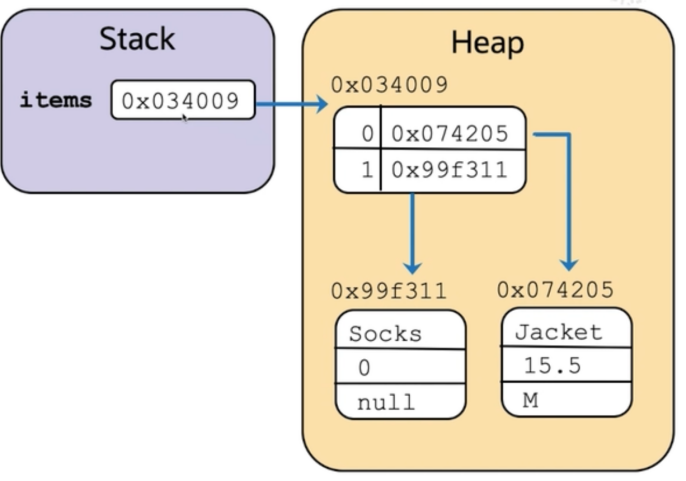
object:



array:



array of object reference:

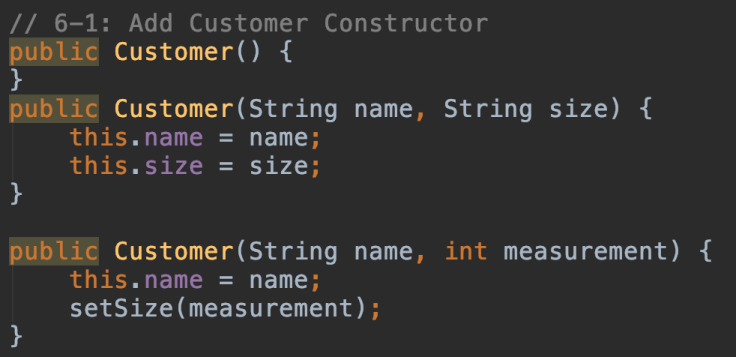


## Constructor

initialization object

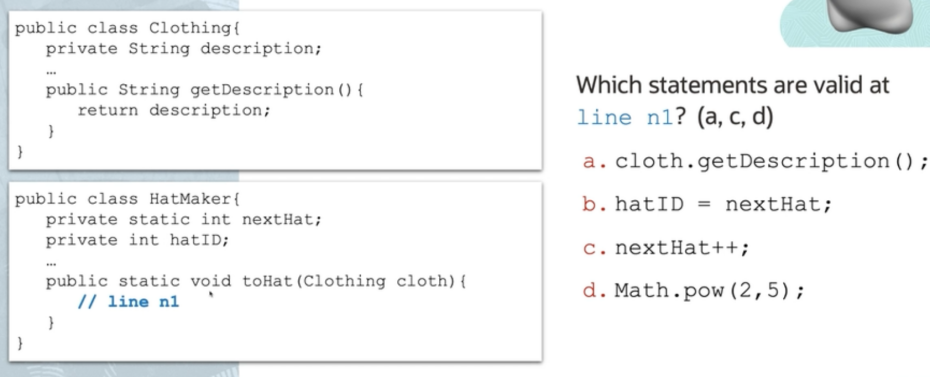
6-1

6-2



## Static

1. belong to the class and is shared by all objects
2. is not specific to any object instance
3. **not** need to initialize objects when use it
4. static method can not see instance variables
5. static method can see static variables



exercise 6-3

# Exceptions

exception types

exception methods

exercise7-1

# OO Approach

## Inheritance

extends

extends **Object** by default

override