Tinker Academy

AP Computer Science Prep(Java Programming)
Lecture 4 - Java Fundamentals 1 (Advantages to OOP)

Winter 2015

What’s special about OOP

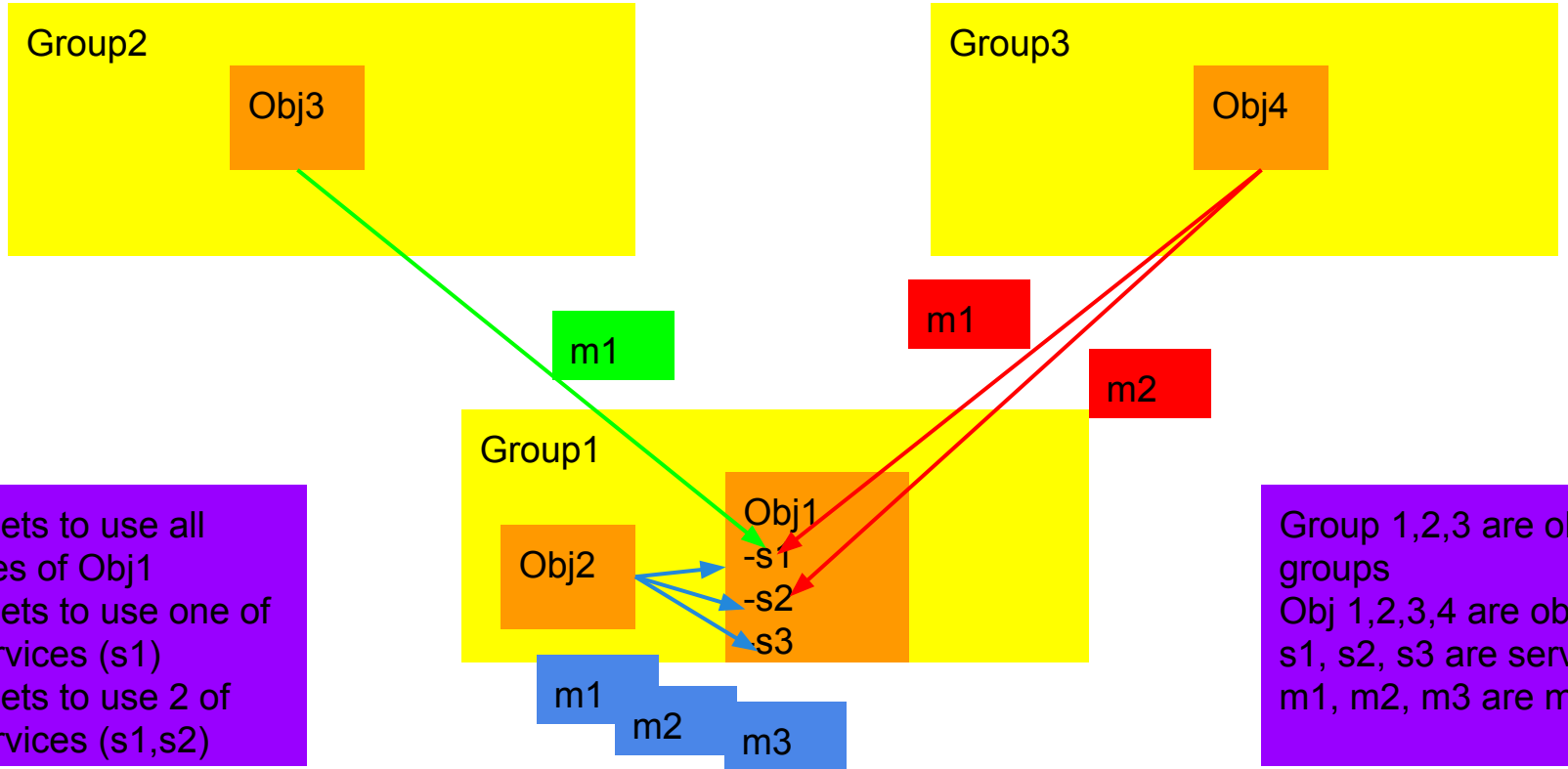
Lecture 4 - Java Fundamentals 1

OOP Provides 3 Advantages

- Support for encapsulating changes
- Support for reusing code
- Support for building interchangeable components
  - OOP provides the ability to interchange objects.
  - This allows building frameworks such as Minecraft and supports swapping components in and out

OOP Advantage #1
Encapsulate Changes

Lecture 4 - Java Fundamentals 1



Obj2 gets to use all services of Obj1
Obj3 gets to use one of the services (s1)
Obj4 gets to use 2 of the services (s1,s2)

Group 1,2,3 are object groups
Obj 1,2,3,4 are objects
s1, s2, s3 are services
m1, m2, m3 are messages

Lecture 4 - Java Fundamentals 1

OOP Advantage #1 - Encapsulation

- No “real world” program is written once and considered “done”
- All “real world” programs evolve constantly.
- New code gets added, some old code is replaced with “better” code.
- A good program is one that is designed to make these changes “easier”

Lecture 4 - Java Fundamentals 1

OOP Advantage #1 - Encapsulation

- OOP provides support for “encapsulation”
- Encapsulation allows code to be changed more easily
- OOP supports this by allowing classes to be grouped together in various ways
- These groups will share some common methods that no other class can share

Lecture 4 - Java Fundamentals 1

OOP Advantage #1 - Encapsulation

- OOP allows you, the programmer to model the program as groups of classes
- In Java these groups are called the class hierarchy and the package
- A class can give groups access to special methods so that later, any changes to the class will impact only those groups
- This may look trivial but is fundamentally important and is calle "encapsulation"

OOP Advantage #2
Reuse Code

Lecture 4 - Java Fundamentals 1

Class A reuses code from its parent class B which reuses code from its parent class C which reuses from the special Object class (ancestor of all classes)

Lecture 4 - Java Fundamentals 1

OOP Advantage #2 - Reuse Code

- OOP also allows code reuse
- A typical program could be several hundreds of thousands of lines long
- Code reuse allows a single class to be written once, tested well and then reused in other parts of the program

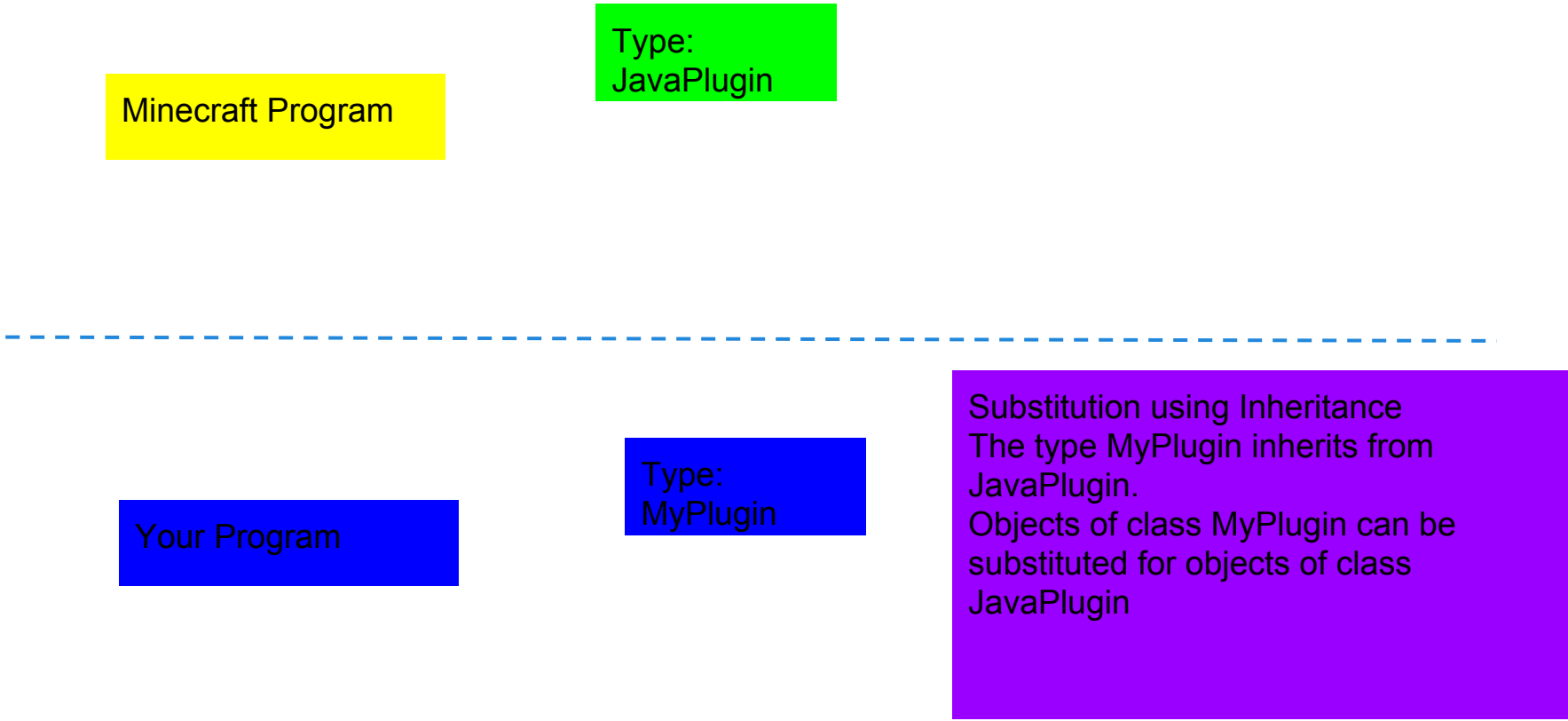
Lecture 4 - Java Fundamentals 1

OOP Advantage #2 - Reuse Code

- OOP supports code reuse by allowing a class to reuse code from another “class”
- This type of code reuse is called “class based inheritance”
- In the diagram, class A inherits code from class B, which inherits code from class C and so on until the special Object class

OOP Advantage #3
Substitute Objects

Lecture 4 - Java Fundamentals 1



Lecture 4 - Java Fundamentals 1

Minecraft Program

Type:
Player

Your Program

Type:
MyPlayer

Substitution using Interfaces
Player is an interface.
MyPlugin shares a common interface with Player
Objects of class MyPlayer can be used wherever Player is used

Lecture 4 - Java Fundamentals 1

OOP Advantage #3 - Substitute Objects

- Programs such as Minecraft are written in Java
- They provide the ability for programmers like you to add new features to the game
- OOP supports this by allowing you, the programmer to define new classes that are based on Minecraft classes or Minecraft interfaces
- Your program objects can now be used in place of those Minecraft classes or interfaces

Class Activity

Lecture 3 - Java Fundamentals 1

Getting Started

- Open File Manager
- Navigate to starterpack/starterpack3
- Click on HelloOOP2.zip
- Right Click, Extract Here
- Start Eclipse
- Import HelloOOP2 into Eclipse

Lecture 3 - Java Fundamentals 1

Getting Started

- Right Click, Properties, Java Build Path
- Add External JARs...
- Select hellopolymorphism.jar
- Click OK

Lecture 3 - Java Fundamentals 1

Add code to main method

- Navigate to HelloOOP2.java under src
- Double click to open
- Type in the following code within the main method { and }

```
CarGenerator generator = new CarGenerator();
for (int i = 0; i < 100; i++) {
        Car car = generator.generateCar();
}
```

- Edit->Select All, Right Click in Editor
- Source->Correct Indentation, File->Save All
- Run, Test Program

Lecture 3 - Java Fundamentals 1

What does this mean?

- hellopolymorphism.jar is a library
- We need to generate Cars from the library using generateCar
- generateCar returns a new Car, we don't care which one.
- All we care is it returned some object (of some class) that implements the Car interface
- Extremely useful concept called polymorphism
- Basis of all java libraries including Minecraft, Facebook Android Libraries