

INF 102

CONCEPTS OF PROG. LANGS

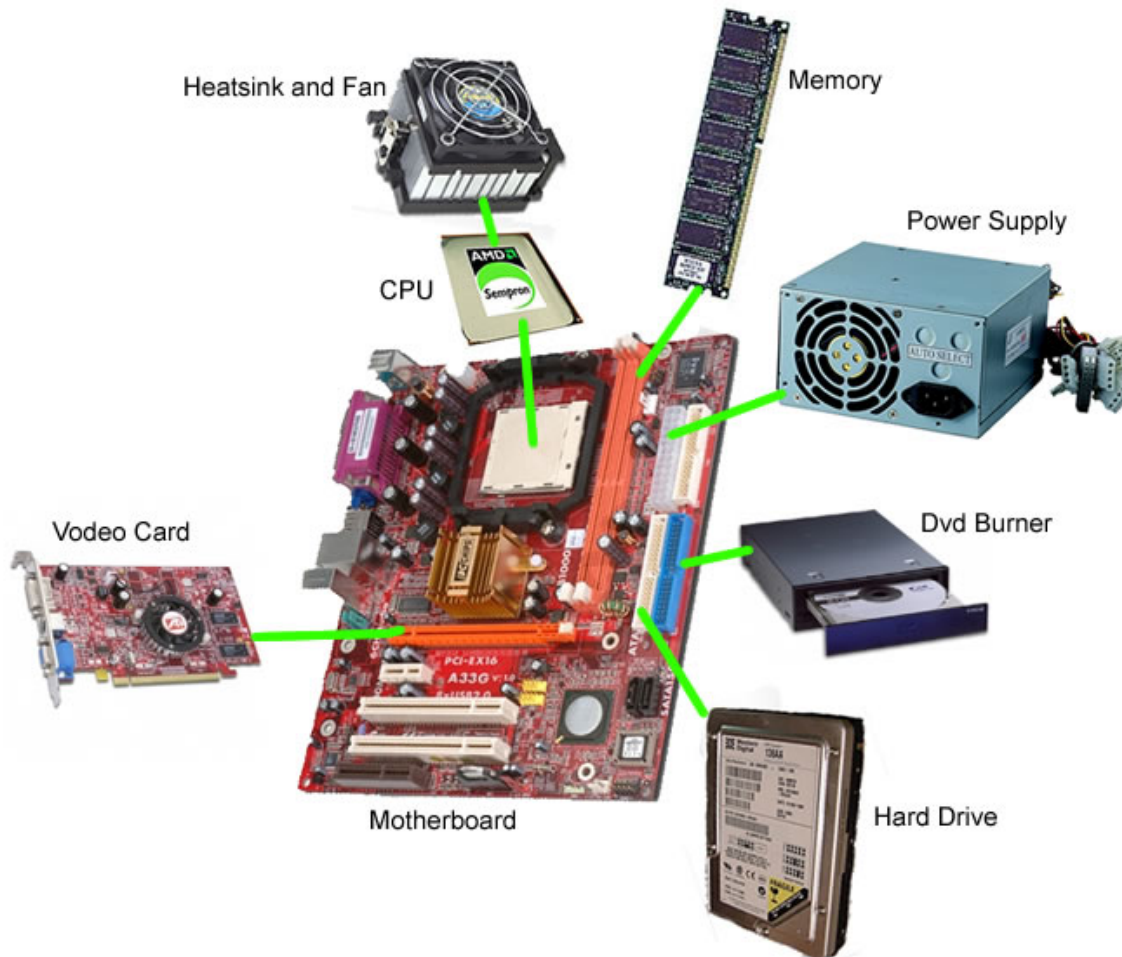
*PLUGINS*

Instructors: James Jones  
Copyright © Instructors.

# Plugins for Evolution&Customization

- Developing software for extension, at a later point in time
  - ▣ By others
  - ▣ By yourself
- Creates new challenges
  - ▣ Potential for “configuration hell”
  - ▣ Combinatorial number of combinations to test and debug

# Modules as physical components



# Lots of Examples

---

- Mozilla Firefox
- Chrome
- Eclipse
- Skinnable Interfaces
- ...

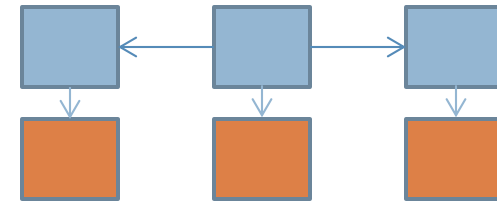
# Software modules as physical components

- Source components
  - Get the source, make it yours. Simple.
- Binary components
  - Java: jar files
  - .NET: DLL files
  - C/C++: so files
  - ...
  - Not so simple

# Linking binary components

## □ 3 steps

1. Independent compilation



2. Dynamic Loading



3. Instantiation of classes

# Linking binary components

---

- Dynamically-typed languages
  - ▣ Simple
- Statically-typed languages
  - ▣ Not so simple

# Binary components -- Python

```
1 #!/usr/bin/env python
2 import sys, ConfigParser, imp
3
4 def load_plugins():
5     config = ConfigParser.ConfigParser()
6     config.read("config.ini")
7     words_plugin = config.get("Plugins", "words")
8     frequencies_plugin = config.get("Plugins", "frequencies")
9     global tfwords, tffreqs
10    tfwords = imp.load_compiled('tfwords', words_plugin)
11    tffreqs = imp.load_compiled('tffreqs', frequencies_plugin)
12
13 load_plugins()
14 word_freqs = tffreqs.top25(tfwords.extract_words(sys.argv[1]))
15
16 for (w, c) in word_freqs:
17 print w, ' - ', c
```



# Binary components – Python

---

- Python
  - ▣ No need to worry about types during independent compilation

# Binary components – Typed

```
class TFApp {  
  static void main(String[] args) {  
    HashMap<String, int> wordFreqs;  
    wordFreqs = tffreqs.top25(tfwords.extract_words(sys.argv[0]));  
  }  
}
```

Types?



# Binary components – Typed

```
interface TFWords {
    public List<String> extractWords(string path);
}

interface TFFreqs {
    public HashMap<String, int> top25(List<String> words);
}

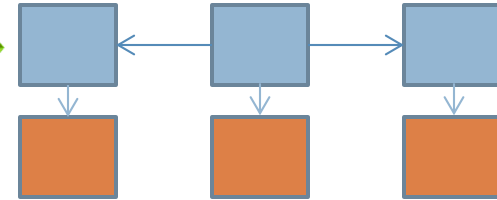
class TFApp {
    static void main(String[] args) {
        HashMap<String, int> wordFreqs;
        TFWords tfwords; // = ???
        TFFreqs tffreqs ; // = ???
        wordFreqs = tffreqs.top25(tfwords.extract_words(sys.argv[0]));
    }
}
```

**Types ok!**

# Linking binary components

## □ 3 steps

### □ Independent compilation



### □ Dynamic Loading



### □ Instantiation of classes

# Binary components – Typed

```
interface TFWords {
    public List<String> extractWords(string path);
}
interface TFFreqs {
    public HashMap<String, int> top25(List<String> words);
}
```

```
class TFApp {
    static void main(String[] args) {
        HashMap<String, int> wordFreqs;
        TFWords tfwords; //= ???
        TFFreqs tffreqs ;//= ???
        wordFreqs = tffreqs.top25(tfwords.extract_words(sys.argv[0]));
    }
}
```

# Binary components – Typed

```
class TFApp {
  static void main(String[] args) {
    HashMap<String, int> wordFreqs;
    TFWords tfwords = new Words1();
    TFFreqs tffreqs = new Freqs1();
    wordFreqs = tffreqs.top25(tfwords.extract_words(sys.argv[0]));
  }
}
```

??????

Coupling between physical components!  
TFApp needs one of the other components  
in order to **compile!**

# Binary components – Typed

```
class TFApp {
  static void main(String[] args) {
    HashMap<String, int> wordFreqs;
    TFWords tfwords = create instance dynamically ("...");
    TFFreqs tffreqs = create instance dynamically ("...");
    wordFreqs = tffreqs.top25(tfwords.extract_words(sys.argv[0]));
  }
}
```

You need to research how to do it  
in your language of choice

Given by .ini file

# Dynamic loading of libraries

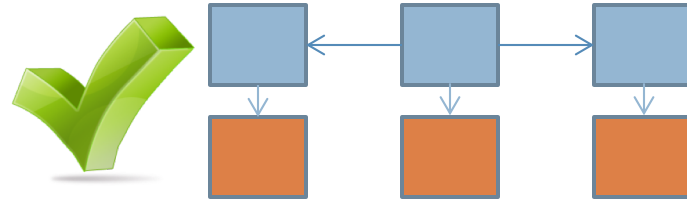
- Python
  - imp module
- Java:
  - ClassLoader class
- .NET
  - Assembly class (Activator)
- Raw C++, Linux:
  - dlopen, dlsym, dlclose
- Raw C++, Win32
  - LoadLibrary()



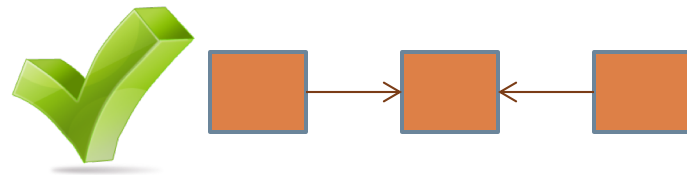
# Linking binary components

## □ 3 steps

□ Independent compilation



□ Dynamic Loading



□ Instantiation of classes

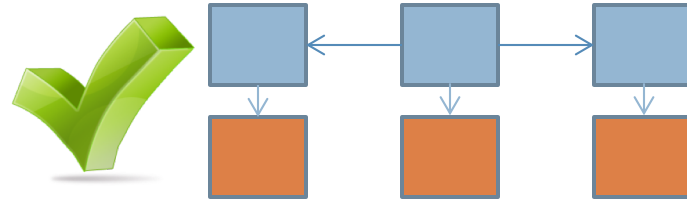
# Instantiation

- Python
  - ▣ `imp.load_compiled("...")`
- Java
  - ▣ `Class.forName("...").newInstance();`
- .NET
  - ▣ `Activator.CreateInstance(type)`

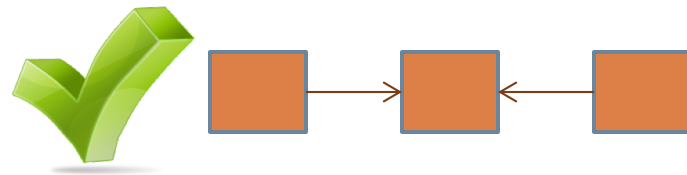
# Linking binary components

## □ 3 steps

□ Independent compilation



□ Dynamic Loading



□ Instantiation of classes

