PROGRAMMING LANGUAGES RECITATION

-Monika Dagar

17th September 2020

Agenda

- Binding
- Scoping
- Lifetime
- Short Circuit Evaluation

Binding

```
int i = 1 // Name Value
void my_func() // Name
{ function
}
```

- An association of entities with identifiers (name).
- Entities: classes, functions, data, types, etc.
- Binding Time - The time at which the association is made.
- Static Binding (Early Binding) – Performed before the program is running (at compile time)
- Dynamic Binding (Late Binding) – Performed during the program execution (during run time).

Advantages of Static Binding:

- Efficiency: the association are made during the compilation time, which means the compiler could make some optimizations for the code generation
- Invariance: the compilation phase fixes all types of variables and expressions.

Advantages of Dynamic Binding:

- Flexibility: languages give more control to programmer (e.g. postpone binding)
- Polymorphic code: code that can be used on objects of different types is polymorphic.

O/P ?

Java

```
public class NewClass {
    public static class superclass {
        static void print()
        {
            System.out.println("print in superclass.");
        }
    }
    public static class subclass extends superclass {
        static void print()
        {
            System.out.println("print in subclass.");
        }
    }

    public static void main(String[] args)
    {
        superclass A = new superclass();
        superclass B = new subclass();
        A.print(); → Print in superclass
        B.print(); → Print in superclass
    }
}
```

Java

```
public class NewClass {
    public static class superclass {
        void print()
        {
            System.out.println("print in superclass.");
        }
    }

    public static class subclass extends superclass {
        @Override
        void print()
        {
            System.out.println("print in subclass.");
        }
    }

    public static void main(String[] args)
    {
        superclass A = new superclass();
        superclass B = new subclass();
        A.print();
        B.print();
    }
}
```

→ Print in superclass

→ Print in subclass

Scoping

- Determine the visibility of an entity in a program, such as a variable.
- **Scope:** The portion of the program where a particular binding is active or visible.
- Global scope: binding is visible throughout the entire programs.
  - *Global variable: A variable with global scope. The lifetime of this kind variable is the duration from the program started to terminated.*
- Function scope: The scope of variable is within the function.
  - *Local variable: a variable with function scope. The lifetime is the duration of the function call.*
- Variable shadowing: In a certain scope, if you redeclare a variable, the original binding is hidden, and has a hole in its scope.

Static/**Lexical Scoping**

- Def. binding of a name is determined by rules that refer only to the program text.
- Thus, the scope of a variable depends on the code (syntactic) structure.
- **Most languages use some variant of this**

Dynamic Scoping

- Binding of a name is given by the most recent declaration encountered at runtime.
- That means: the variable's scope is depending on the execution order.
- Very few languages implement dynamic scoping
- Used in Lisp, Snobol, APL

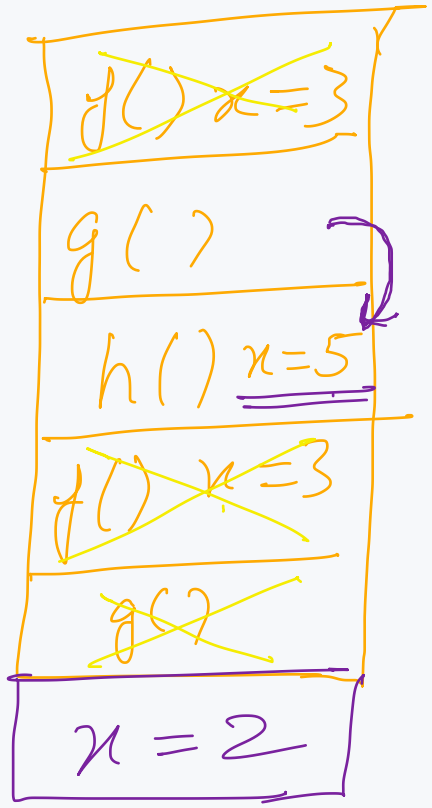
```c
int x = 2; // This is not a global variabe

void f(){
    int x = 3;
}

int g(){
    f();
    return x + 4;
}

int h(){
    int x = 5;
    return g();
}

printf("function g returns %d", g());
printf("function h returns %d", h());
```



Static

6

6

Dynamic

6

9

Lifetime

- The period of time between the creation of an entity and its destruction.
- Scope is a place(or many places), whereas lifetime is a time span.

- Memory Allocation:
  - Static
  - Stack
  - Heap

Short Circuit Evaluation

- An expression is stopped being evaluated as soon as its outcome is determined.

```
if (a==b || c==d)
{
   ==
}
```

if 'a==b' is true, 'c==d' will not be executed

```
if (a==b && c==d)
{
   ——
   ——
}
```

if 'a==b' is false, 'c==d' will not be executed

```
=> if (a != null && a.getValue()==45)
     {  ——  }
```