Offline Application Security Testing:
Static Analysis

Lecture-11

Department of CSE, IUT

Outline
s

- Program Analysis Basic
- Static vs Dynamic Analysis
- Static Analysis: The Big Picture
- Inside a Static Analysis Tool
- Hands on Exercises

Program Analysis Basics

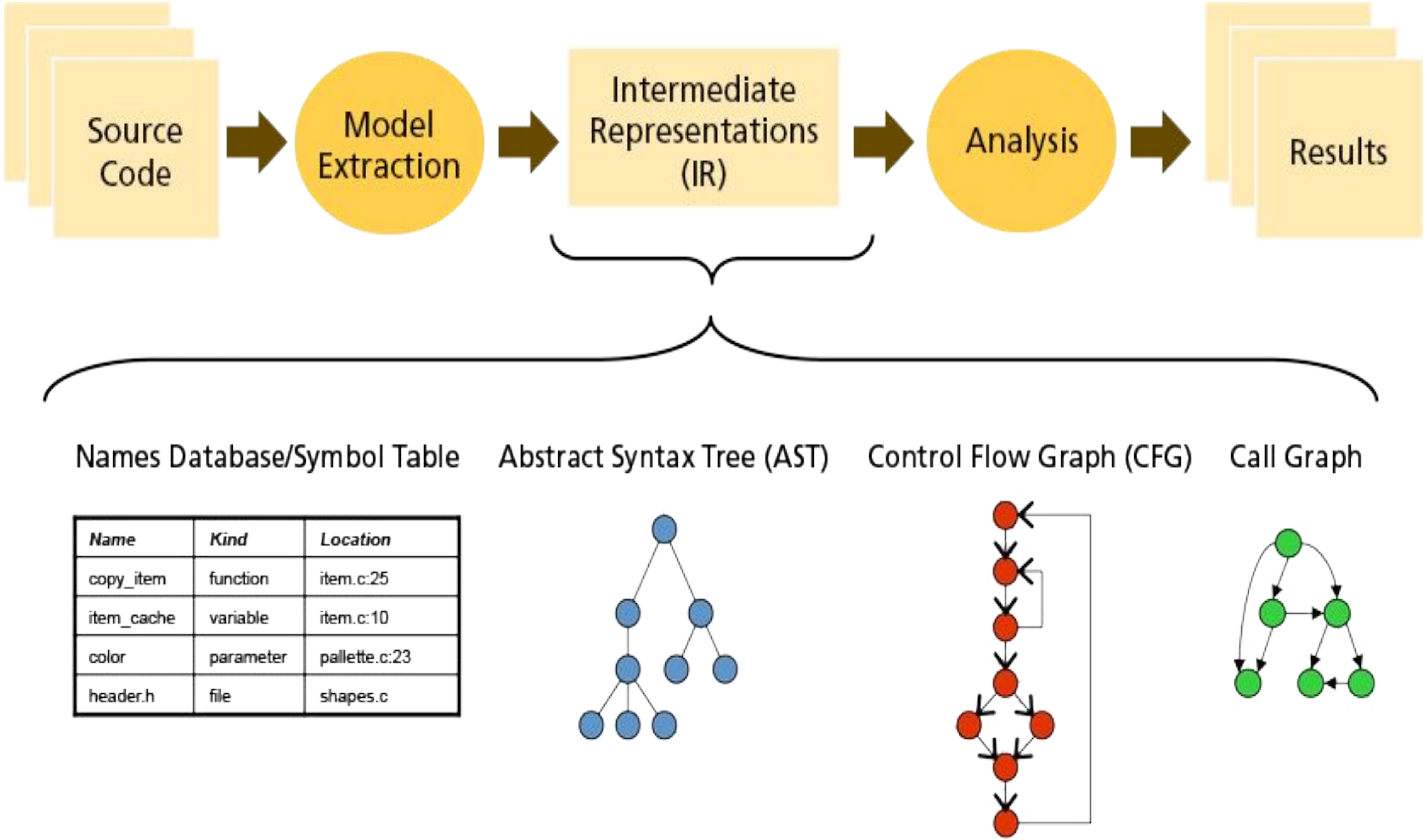
- Analyze a program for potential bugs
- Often source cannot be directly analyzed
- Source code is converted to some intermediate form (object code, bytecode)
- Use sophisticated tools to explore program statements, paths and branches to find potential bugs or inconsistencies.

Static vs Dynamic Analysis

- **Static Analysis** deals with source code and its variants
  - Do not run the code
  - Explore all branches
  - Lot of False Positives

- **Dynamic Analysis** run the program and see what it is doing.
  - Can explore one path at a time
  - Success depends on input generation and path exploration

Static Analysis: The Big Picture



Source Code

Model Extraction

Intermediate Representations (IR)

Analysis

Results

Names Database/Symbol Table

Abstract Syntax Tree (AST)

Control Flow Graph (CFG)

Call Graph

Name	***Kind***	***Location***
copy_item	function	item.c:25
item_cache	variable	item.c:10
color	parameter	pallette.c:23
header.h	file	shapes.c

Static Code Checking Tasks

- Type checking
- Style checking
- Program understanding
- Program verification / Property checking
- Bug finding
- Security review

Secure Programming with Static Analysis

- The line between secure/insecure is often subtle

- Many seemingly non-security decisions affect security

- Small problems can hurt a lot

- Smart people make dumb mistakes

Common Software Bugs

Generic Mistakes

- Input validation
- Memory safety (buffer overflow)
- Handling errors and exceptions
- Maintaining privacy

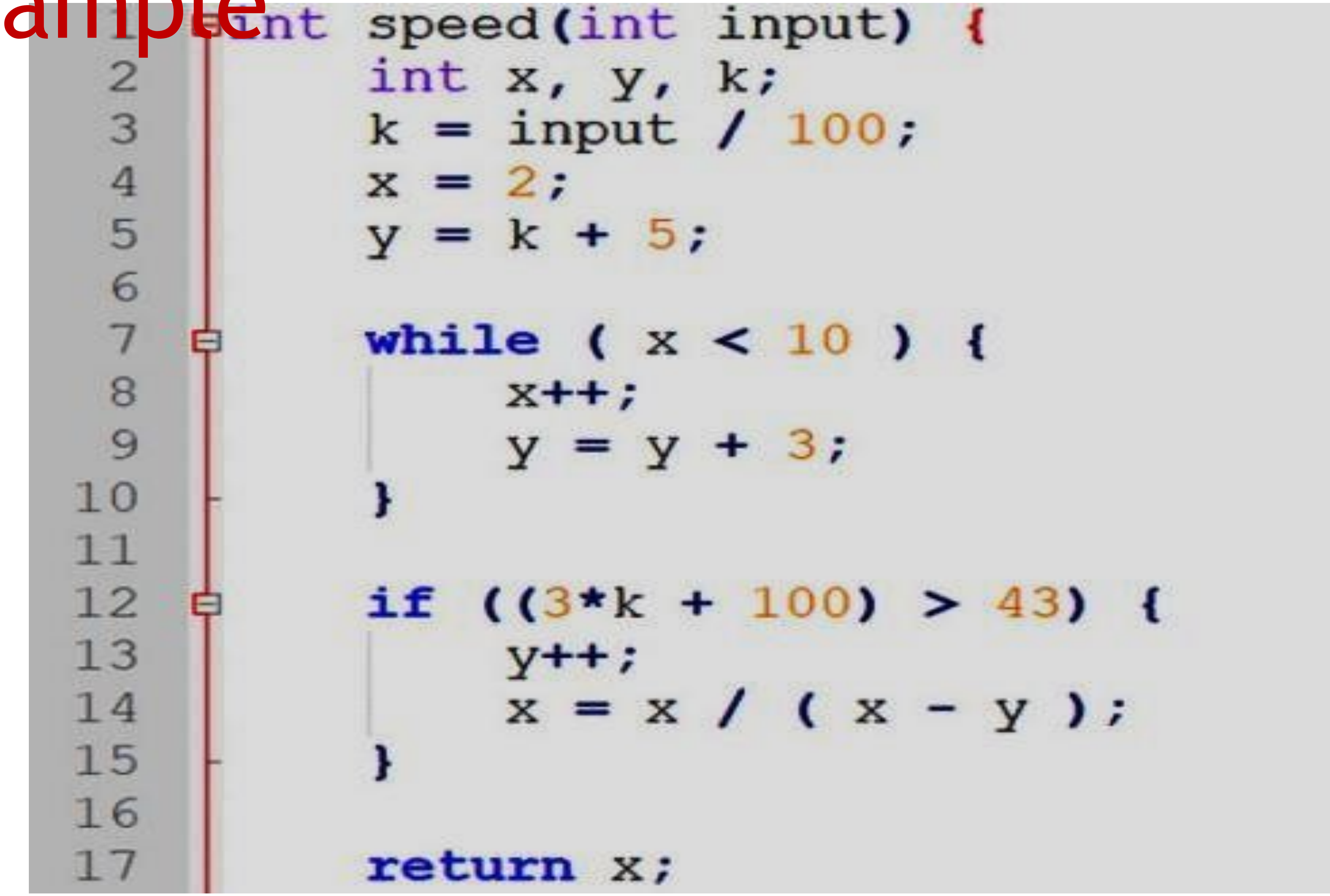
Common Software Varieties Web

- applications Network
- services Privileged
- programs

An Example

```
int main(int argc, char* argv[]) {
    char buf1[1024];
    char buf2[1024];
    char* shortString = "a short string";
    strcpy(buf1, shortString); /* eh. */
    strcpy(buf2, argv[0]);     /* !!! */
    ...
```

Another Example



```
int speed(int input) {
    int x, y, k;
    k = input / 100;
    x = 2;
    y = k + 5;

    while ( x < 10 ) {
        x++;
        y = y + 3;
    }

    if ((3*k + 100) > 43) {
        y++;
        x = x / ( x - y );
    }

    return x;
```

Behind the Scene of a Static Analyzer

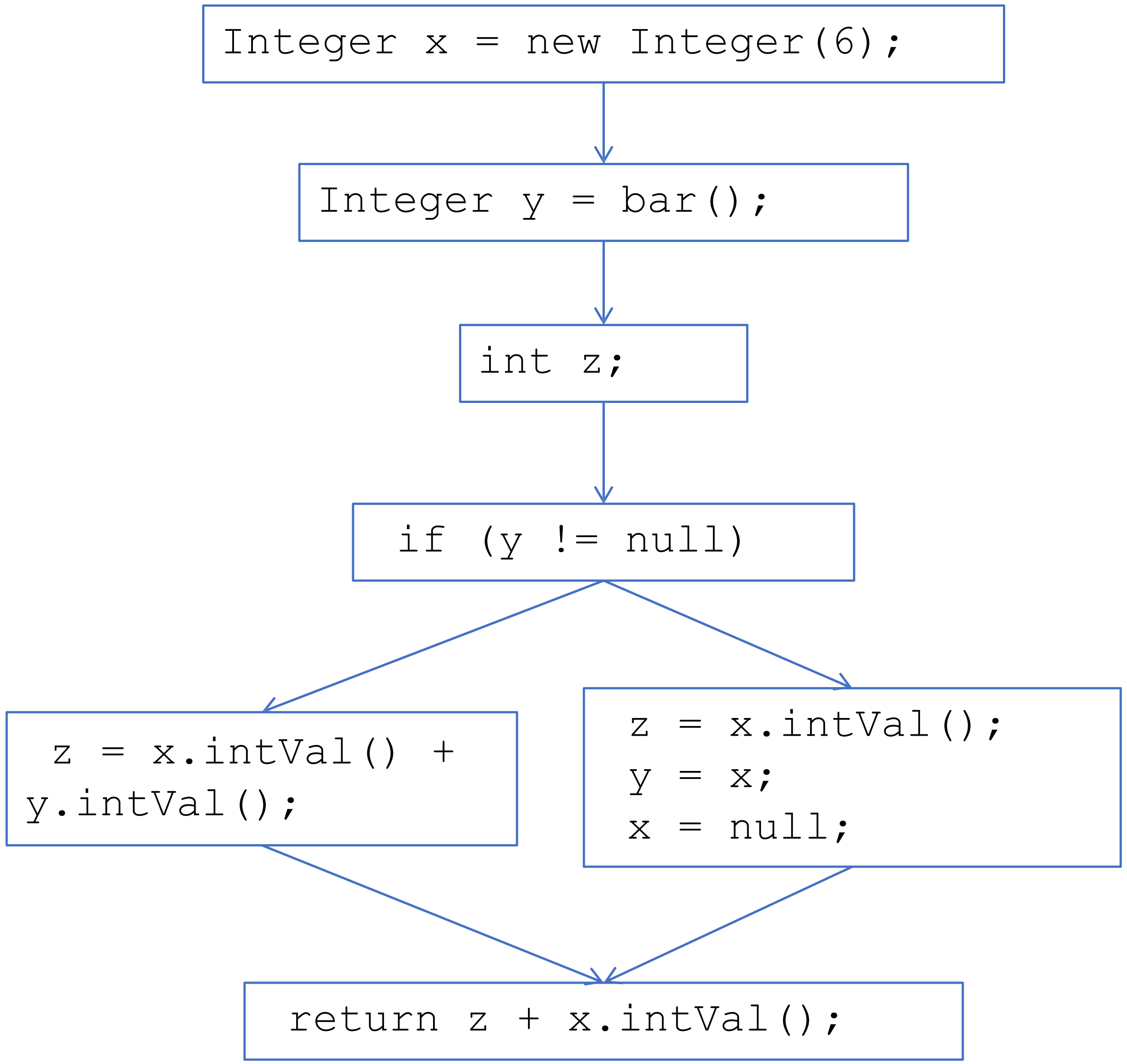
Exercise time!

```
int foo() {
    Integer x = new Integer(6);
    Integer y = bar();
    int z;
    if (y != null)
        z = x.intVal() + y.intVal();
    else {
        z = x.intVal();
        y = x;
        x = null;
    }
    return z + x.intVal();
}
```

Are there any possible **null pointer exceptions** in this code?

In graph form…

```
int   foo()     {
        Integer  x    =    new      Integer(6);
        Integer  y    =    bar();
        int    z;
        if      (y   !=   null)
             z=   x.intVal()     +    y.intVal();
        }else     {
             z=   x.intVal();
             y=   x;
             x=   null;
 }
 return   z    +    x.intVal();  }
```



```
Integer x = new Integer(6);
Integer y = bar();
int z;
if (y != null)
```

```
z = x.intVal() +
y.intVal();
```

```
z = x.intVal();
y = x;
x = null;
```

```
return z + x.intVal();
```

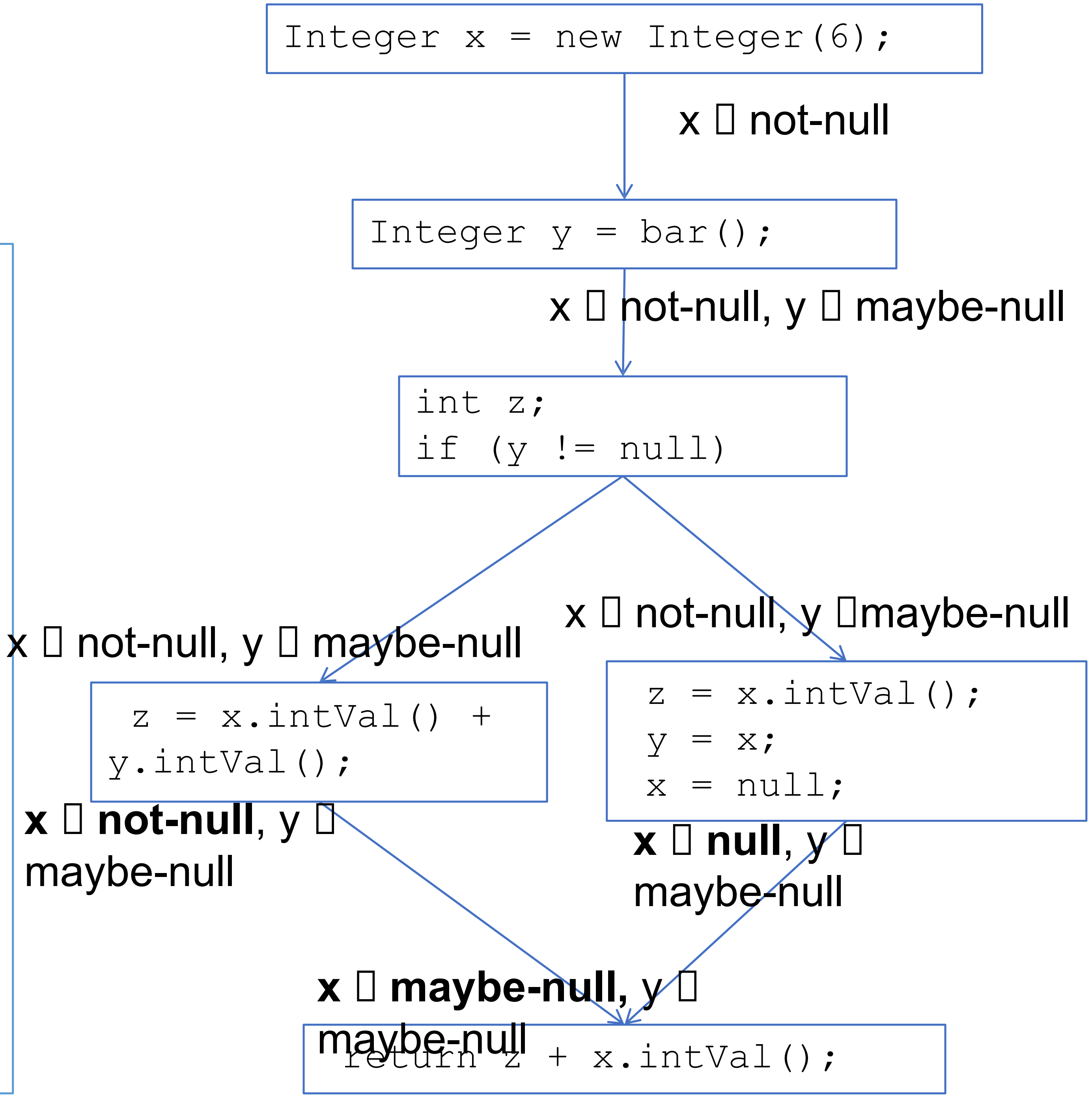
Goal : Null pointer analysis

- Track each variable in the program at all program points.
- Abstraction:
  - Program counter
  - 3 states for each variable: null, not-null, and maybe-null.
- Then check if, at each dereference, the analysis has identified whether the dereferenced variable is or might be null.

In graph form…

```
1.int    foo()
{
2.      Integer   x    =    new      Integer(6);
3.      Integer   y    =    bar();
4.      int     z;
5.      if (y   !=   null)
6.            z =    x.intVal()    +    y.intVal();
7.      } else      {
8.            z =    x.intVal();
9.            y =    x;
10.           x =    null;
11. }
12. return   z    +    x.intVal();  13.}
```

Error: may have null pointer on line 12, because x may be null!



```
Integer x = new Integer(6);
```

x not-null

```
Integer y = bar();
```

x not-null, y maybe-null

```
int z;
if (y != null)
```

x not-null, y maybe-null

x not-null, y maybe-null

```
 z = x.intVal() +
y.intVal();
```

```
 z = x.intVal();
 y = x;
 x = null;
```

x not-null, y maybe-null

x null, y maybe-null

x maybe-null, y maybe-null

```
 return z + x.intVal();
```

Lets Try Some Examples

1. **Online Static Analyzer:**

   https://www.gimpel.com/demo.html

2. **Find some Examples of Vulnerable Codes**

References

- Lecture by Dr. Sarker Tanveer Ahmed on workshop titled “Hands on Training on Fundamental Web and Application Security Issues for NREN Professionals” organized by Institute of Information Technology (IIT) University of Dhaka.
- https://owasp.org/www-community/controls/Static_Code_Analysis