Department of Computer Engineering
University of Peradeniya
CO523 - Programming Languages

Lab 05: Data Types, Type Systems, and Type Checking

Objectives

- Understand and classify different data types in programming languages.
- Explain the purpose of a type system and its role.
- Distinguish between static vs dynamic typing and strong vs weak typing, and understand how they affect program behaviour.
- Apply typing concepts practically by writing, running, and debugging programs.
- Analyze program outputs and errors to understand type enforcement, implicit/explicit conversions, and runtime behaviour.

1. Data Types

A **data type** defines a collection of data values and a set of predefined operations on those values. Computer programs produce results by manipulating data. An important factor in determining the ease with which they can perform this task is how well the data types available in the language being used match the objects in the real world of the problem being addressed. Therefore, it is crucial that a language supports an appropriate collection of data types and structures.

Data types are commonly categorized into three main groups: **Primitive data types, Composite data types,** and **User-defined data types**. Primitive data types represent basic, built-in values such as integers, floating-point numbers, characters, and boolean values. Composite data types combine multiple primitive values into more complex structures, such as arrays, structures, and Classes, enabling the modelling of more sophisticated data relationships. User-defined data types allow programmers to create new types tailored to specific problem domains, thereby improving the abstraction, readability, and maintainability of programs.

2. Type Systems

A **type system** is a set of types and the rules that govern their use in programs. Obviously, every typed programming language defines a type system. The formal model of a type system of a programming language consists of a set of types and a collection of functions that define the type rules of the language, which are used to determine the type of any expression.

A type system helps detect errors early, enforce correctness, improve documentation, enable compiler optimizations, and increase overall program safety.

3. Static vs Dynamic typing

In computer programming, languages can be classified based on when types are checked, as **statically typed** or **dynamically typed** languages. At its core, this distinction refers to whether a programming language determines and enforces the type of a variable at compile time or at runtime, which affects error detection, flexibility, and program behaviour.

3.1. Static typing

Static typing is a characteristic of a programming language in which the type of every variable is determined and checked at compile time. Operations involving incompatible types are detected before the program runs, preventing meaningless operations and reducing the risk of runtime errors. This early enforcement improves program reliability, increases safety, and enables compiler optimizations.

Java is a statically typed language. Consider the following code and try running it:

```
public class Main {
    public static void main(String[] args) {
        int x;
        x = "hello";
        System.out.println(x);
    }
}
```

When you try to compile this code, the compiler raises an error because "hello" is a string, not an integer. The type mismatch is caught before the program runs, demonstrating static typing.

3.2. Dynamic typing

Dynamic typing is a characteristic of a programming language in which the type of a variable is determined and checked at runtime. In these systems, variables can hold values of different types at different times, and operations on incompatible types are only detected when the program executes. This flexibility allows for faster development and less verbose code, but it can increase the risk of runtime errors and reduce execution performance compared to statically typed languages.

Python is an example of a dynamically typed language. Refer to the following example using python:

```
x = 10
x = "hello"
print(x)
```

When you run this code, it executes without errors and prints "hello". The variable x changes from an integer to a string at runtime, demonstrating dynamic typing.

4. Strong vs Weak typing

Programming languages can also be classified based on how they handle data types, as **strongly typed** versus **weakly (or loosely) typed** languages. At its core, this distinction refers to how strictly a programming language distinguishes between different data types and how rigorously it enforces its type system.

4.1. Strong Typing

Strong typing refers to a characteristic of a programming language in which the type system is strictly enforced. In such languages, operations involving incompatible data types are not permitted unless an explicit type conversion is performed. This strict enforcement prevents implicit type coercion and ensures that type-related errors are detected rather than silently ignored, thereby improving program reliability and correctness.

Python is a strongly typed language. Consider the following code and try running it to observe the results:

```
x = "5"
y = 3
print(x + y)
```

When you run this code, Python raises a TypeError because it does not allow adding a string ("5") and an integer (3) directly. This behavior demonstrates strong typing: incompatible types cannot be combined without explicit conversion.

4.2. Weak Typing

Weak typing refers to a characteristic of a programming language in which the type system is enforced less strictly. In such languages, operations involving incompatible data types are often permitted through implicit (automatic) type conversion, allowing the language to coerce values into compatible forms in order to complete the operation. While this flexibility can simplify programming, it may also allow type-related errors to occur at runtime or remain undetected.

JavaScript is a best example for weakly typed language. Try running the following JavaScript Code to see the result:

```
let x = "5";
let y = 3;
console.log(x + y);
```

When you run this code, JavaScript outputs "53". The language automatically converts the number 3 to a string and concatenates it with "5". This demonstrates weak typing: the language allows implicit type coercion.

Lab Tasks

Task 01

Run the following Python script and observe the output, including the error messages.

```python
data = [1, "2", 3.0, "hello"]

for i in range(len(data)):
    for j in range(i + 1, len(data)):
        try:
            result = data[i] + data[j]
            print(f"{data[i]} + {data[j]} = {result} (type: {type(result)})")
        except TypeError as e:
            print(f"Cannot add {data[i]} ({type(data[i])}) + {data[j]}
({type(data[j])}): {e}")
```

Write a detailed explanation of the results produced by the program. Using the output of the program as evidence, explain whether Python should be classified as a statically typed or dynamically typed language, and whether it should be considered strongly typed or weakly typed. Your explanation must justify both classifications by referring explicitly to the observed output, including which operations succeed, which fail, and the TypeError messages that are produced during execution.

Task 02

Consider the following JavaScript program:

```javascript
let a = 10;
let b = "20";
let c = 3.5;

let result1 = a + b;
let result2 = a + c;
let result3 = b + c;

console.log(result1);
console.log(result2);
console.log(result3);
```

Run the program and record the output. Modify the program to also display the data type of each result. Using your observed results as evidence, explain whether JavaScript should be classified as a statically typed or dynamically typed language, and whether it should be

considered strongly typed or weakly typed. In your explanation, identify which values are implicitly converted in each operation, specify the resulting data types, and discuss whether these behaviors represent advantages or disadvantages in practical programming contexts.

Task 03

Consider the following two programs, one written in Python and the other in Java.

Python:

```python
values = [10, "5", 2.5]

for i in range(len(values)):
    for j in range(i + 1, len(values)):
        try:
            result = values[i] + values[j]
            print(values[i], "+", values[j], "=", result, "(type:",
type(result), ")")
        except TypeError as e:
            print("Error:", e)
```

Java:

```java
public class TypeTest {
    public static void main(String[] args) {
        int a = 10;
        String b = "5";
        double c = 2.5;

        // Uncomment each line one at a time and observe compiler behavior
        // int r1 = a + b;
        // double r2 = a + c;
        // String r3 = b + c;

        String r4 = a + b;
        double r5 = a + c;

        System.out.println(r4);
        System.out.println(r5);
    }
}
```

Run the Python program and observe the runtime output and errors. For the Java program, observe which statements fail at compile time and which successfully compile and run. Next, modify both programs to perform explicit type conversions (for example, converting strings to

numbers before addition or casting numeric values to other numeric types) and observe how this affects program behavior. Using all of your observations as evidence, compare Python and Java in terms of static versus dynamic typing and strong versus weak typing. In your explanation, discuss when type checking occurs in each language, how implicit and explicit type conversions are handled, and the advantages and disadvantages of requiring explicit conversion for certain operations.

Task 04

Is the C language statically typed or dynamically typed? Is it a weakly typed language or a strongly typed language? Explain with examples.

Submission

Create a single PDF file containing all your source codes, execution screenshots, and explanations. Name the file CO523_Lab05_EXXYYY.pdf, where EXXYYY represents your E-number. Ensure all code is properly formatted and readable before submission.

Upload your answer sheet to the FEeLS by the given deadline.