Centro de Investigación y de Estudios Avanzados del IPN
Unidad Tamaulipas

Assignment #6

Student: *Luis Alberto Ballado Aradias*
Course: *Tecnologías Computacionales (Sep - Dec 2022)*
Professor: *Dr. Edwin Aldana Bobadilla*
December 1, 2022

Generics in programming

Generic Programming is a style of computer programming in which algorithms are written in terms of types to be specified later that are then instantiated when needed for specific types provided as parameters. This approach permits writing common functions or types that differ only in the set of types on which they operate when used, reducing **duplication**.

Allow functions, methods and classes to work with arguments of any type whilst maintaining the information on the relationships between thigs, such as arguments and return values.

In other words, a **generic** is a function or similar that will work on a variety of input types.

One great example is the List class. When providing type hints for a list class, we can provide the type that should be stored in the class in square brackets []
Advantages of generics

- **Reusability** - You can use a single generic type def inition for multiple purposes in the same code without any alterations. For example, you can create a generic method to add two numbers. This method can be used to add two integers as well as two floats without any modification in the code.
- **Type Safety** - Generic data types provide better type safety, especially in the case of collections. When using generics you need to define the type of objects to be passed to a collection. This helps the compiler to ensure that only those object types that are defined in the definition can be passed to the collection.
- **Performance** - Generic types provide better performance as compared to normal system types because they reduce the need for boxing, unboxing, and typecasting of variables or objects.

Generics in Java

Generics means parameterized types. The idea is to allow type to be a parameter to methods, classes, and interfaces. Using Generics, it is possible to create classes that work with different data types.

Types of Java Generics.

- **Generic Method:** Takes a parameter and returns some value after performing a task.
- **Generic Class:** It is implemented exactly like a non-genetic class, but it contains a type parameter section.

Example in Java (Generic Class).

```
public class Box<T> {
   private T t;

   public void add(T t) {
      this.t = t;
   }

   public T get() {
      return t;
   }

   public static void main(String[] args) {
      Box<Integer> integerBox = new Box<Integer>();
      Box<String> stringBox = new Box<String>();

      integerBox.add(new Integer(10));
      stringBox.add(new String("Hello World"));

      System.out.printf("Integer Value :%d\n\n", integerBox.get());
      System.out.printf("String Value :%s\n", stringBox.get());
   }
}
```

Salida:

```
Integer Value :10
String Value :Hello World
```

Generics in C#

Generic classes and methods combine reusability, type safety, and efficiency in a way that their non-generic counterparts cannot. Generics are most frequently used with collections and the methods that operate on them.

Example in C#.

```
class DataStore<T>
{
    public T Data { get; set; }
}

class KeyValuePair<TKey, TValue>
{
    public TKey Key { get; set; }
    public TValue Value { get; set; }
}


KeyValuePair<int, string> kvp1 = new KeyValuePair<int, string>();
kvp1.Key = 100;
kvp1.Value = "Hundred";
Console.WriteLine(kvp1.Key + ", " + kvp1.Value);

KeyValuePair<int, string> kvp1 = new KeyValuePair<int, string>();
kvp1.Key = 100;
kvp1.Value = "Hundred";
Console.WriteLine(kvp1.Key + ", " + kvp1.Value);
```

Generics in Python

In Python, all variables and all parameters of a function are the names bound to untyped references. The type is bound to the passed object, and it is checked only when the operation with the object is executed.

For example on the calculator proyect I defined the Union of two set as the sum (setA + setB) of the two set insted of using setA.union(setB), because of that, any function in Python is a generic function.

Example in Python.

```
from typing import TypeVar, Generic, List

T = TypeVar('T')

class Stack(Generic[T]):
    def __init__(self) -> None:
        # Create an empty list with items of type T
        self.items: List[T] = []

    def push(self, item: T) -> None:
        self.items.append(item)

    def pop(self) -> T:
        return self.items.pop()

    def empty(self) -> bool:
        return not self.items
```

Salida

```
stack = Stack[int]()
stack.push(2)
stack.pop()
stack.push('x') # as we defined as int instance types error
```

Summary. Generics is a class which allows the user to define classes and methods.
When we are writting a code that should handle different data types, Generics takes an important role and icrease the reusability of the code, and also removes the possibilities of boxing and unboxing code.

References.
https://medium.com/@steveYeah/using-generics-in-python-99010e5056eb
https://www.geeksforgeeks.org/generics-in-java/
https://learn.microsoft.com/en-us/dotnet/csharp/fundamentals/types/generics
https://stackoverflow.com/questions/6725868/generics-templates-in-python
https://www.tutorialspoint.com/java/java_generics.htm