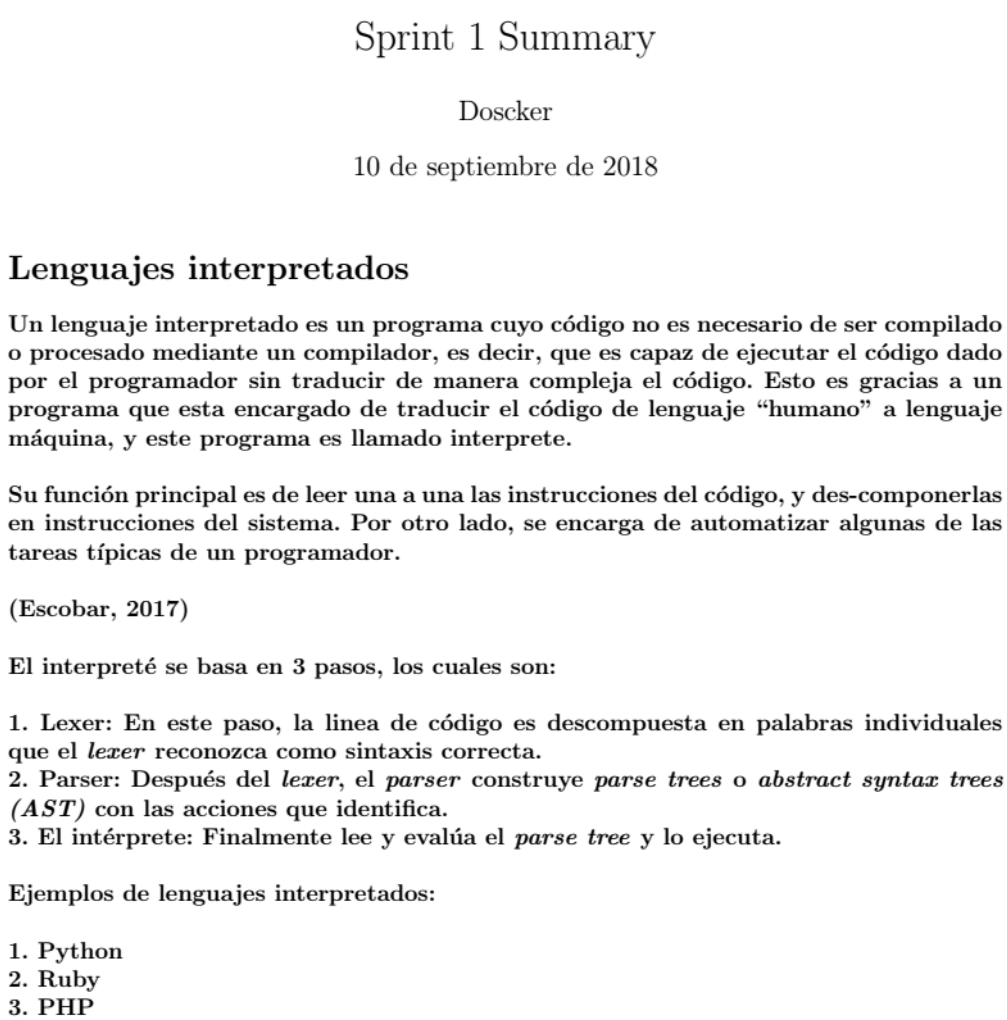
Sprint 2 Summary

Doscker

26 de septiembre de 2018

Journal del trabajo completado completado hasta ahora



Sprint 1 Summary

Doscker

10 de septiembre de 2018

Lenguajes interpretados

Un lenguaje interpretado es un programa cuyo código no es necesario de ser compilado o procesado mediante un compilador, es decir, que es capaz de ejecutar el código dado por el programador sin traducir de manera compleja el código. Esto es gracias a un programa que esta encargado de traducir el código de lenguaje "humano" a lenguaje máquina, y este programa es llamado interprete.

Su función principal es de leer una a una las instrucciones del código, y des-componerlas en instrucciones del sistema. Por otro lado, se encarga de automatizar algunas de las tareas típicas de un programador.

(Escobar, 2017)

El interpreté se basa en 3 pasos, los cuales son:

1. Lexer: En este paso, la linea de código es descompuesta en palabras individuales que el *lexer* reconozca como sintaxis correcta.
2. Parser: Después del *lexer*, el *parser* construye *parse trees* o *abstract syntax trees (AST)* con las acciones que identifica.
3. El intérprete: Finalmente lee y evalúa el *parse tree* y lo ejecuta.

Ejemplos de lenguajes interpretados:

1. Python
2. Ruby
3. PHP

Figura 1: Para ver el Journal completo, visitar: https://github.com/Fernando0107/Doscker/blob/master/Dos

Aprender a profundidad las características de Java

Preguntas de las lecturas:

Diego:

1. ¿Qué pasa cuando creamos una variable?
2. ¿Qué puede pasar si omitimos poner *println*?
3. ¿Cuál es la diferencia entre un double y un int?
4. ¿Por qué es importante esta diferencia?
5. ¿Qué es un método?
6. ¿Se puede llamar a un método más de una vez?
7. ¿Cuál es la diferencia entre una inicialización y una declaración?
8. ¿Para qué se usa la keyword “class”?
9. ¿Cuál es la diferencia entre el + que se utiliza con strings y el que se utiliza con integers?
10. ¿Qué significa concatenar?
11. ¿Qué es bytecode?
12. ¿Qué significa lenguaje de alto-nivel?
13. ¿Es Java un lenguaje de alto nivel?
14. ¿Qué es un operando?
15. ¿Se pueden utilizar los operandos con strings?

Adriana:

1. ¿Qué regla debe seguir Java cuando aparece más de un operador en una expresión?

Figura 2: Para ver el todas las preguntas, visitar: https://github.com/Fernando0107/Doscker/blob/master/Pr

Entender los tipos datos primitivos en Java

Tipos de datos primitios en Java

Doscker Quan, Diego Gonzalez, Fernando
Lindsey, Maite Elgueta, Juan José Mundo, Adriana

September 8, 2018

1 Tipos de datos primitivos

Los tipos de datos definen valors que una variable puede tomar o representar. Actualmente en java se utilizan dos tipos de datos, los primitivos y los no primitivos. Arreglos y cadenas de carácteres son no primitivos mientras que el resto son primitivos.
Java es un lenguaje que es escrito estaticamente, es decir, todas las variables y métodos deben de ser declarados antes de ser compilado. Un ejemplo de como declarar una variable es:

```
int num;
```

Java utiliza estos tipso de datos para poder mantener la portabilidad de los programas escritos en este, ya que estos datos no cambian de tamaño de ordenador a ordenador. Los tipos de datos primitivos se demuestran en la siguiente tabla:

Nombre	Descripción	Tamaño	Rango o Ejemplo
boolean	true or false	1 bit	true, false
byte	númerico entero	8 bits	-128 a 127
char	Unicode character	16 bits	”a”, ”b”
short	númerico entero	16 bits	-32,768 a 32767
int	númerico entero	32 bits	-2,147,483,648 a 2,147,483,647
long	númerico entero	64 bits	-9 billones a 9 billones
float	números fraccionales	32 bits	números de 7 digitos
double	números fraccionales	64 bits	números de 15 digitos

Figura 3: Para ver el documento el PDF sobre los tipos de datos primitivos, visitar: https://github.com/Fernando0107/Doscker/blob/master/datos20primitvos20(1).pdf

Entender el I/O de Java

```java
import java.util.Scanner;


public class Main {
    public static void main(String[] args) {


    Scanner input=new Scanner(System.in);

    String User_input;

    System.out.print("ingrese la palabra");

    User_input=input.next();

    String CAPS;

    CAPS=User_input.toUpperCase();

    System.out.print(CAPS);
    }

}
```

Figura 4: Para ver el documento el código de I/O de Java, visitar: https://github.com/Fernando0107/Doscker/blob/master/IO.java