**Treball de Recerca**

*" Programació de diverses aplicacions per al Sistema Operatiu Android en el llenguatge de programació Java"*

Pablo Fraile Alonso

**Índex:**

**1. Introducció al llenguatge de programació:**

1.1 Definició de programar

1.2 Definició de programa.

1.2.1 Explicació del codi font

1.2.1.1 Explicació de les paraules reservades

1.2.1.2 Explicació de les anotacions

1.2.2 Executable

1.2.3 Explicació de la variant d'un programa anomenat aplicació

1.3 Diferents tipus de llenguatges de programació i la seva utilitat

1.4 Introducció a les llicències dels llenguatges de programació i de software

1.4.1 Llicències de Software Lliure

1.4.2 Llicències de Software Propietari

**2. Compiladors.**

2.1 Definició de Compiladors.

2.2 Tipus de compiladors

2.3 Fases del compilador

**3. Introducció a Java:**

3.1 Definició i característiques del llenguatge de programació Java

3.1.1 Programació dedicada a objectes (POO)

3.2 Introducció a l’ecosistema del codi de java

3.2.1 Java JDK

3.3 Creació de l’aplicació a partir del codi font

3.3.1 Introducció a la compilació del codi en els llenguatges de programació

3.3.2 Compilador adaptat a Java i la JVM

3.4 Java API

**4. Android Studio:**

4.1 Introducció de Android Studio

4.2 Android SDK

Gradle: https://medium.com/@wasyl/understanding-android-gradle-build-files-e4b45b73cc4c

4.3 API packages i classes d'Android

**5. Java Adaptat a Android:**

5.1 Funcionament de Android.

5.2 Android Resources

5.2.1 Fonaments de la creació d'una aplicació Android <https://developer.android.com/guide/components/fundamentals>

https://developer.android.com/guide/practices/screens\_support

5.2.2Introducció del llenguatge XML adaptat a Android Studio

5.2.3 Icones (mesures, adaptació a les noves versions de Android)

5.2.4 Notificacions ( Barra d’estat, pantalla de bloqueig i app icon badge)

**1. Introducció al llenguatge de programació:**

**1.1 Definició de programar:**

El verb programar es refereix a l'art de fer que un dispositiu electrònic ( com per exemple un telèfon mòbil ) dugui a terme unes determinades accions que l'usuari anomenat programador li ha encomanat anteriorment. Hi ha diferents tipus de llenguatges de programació que poden oferir més accions que altres, però finalment tots tenen com a funció realitzar accions encomanades anteriorment pel programador.

**1.2 Definició de programa:**

Per a poder definir la paraula programa, s'ha decidit citar la definició que dona la Universitat Internacional d' Atlanta :

"Un programa es un conjunt d'instruccions o ordres basades en un llenguatge de programació que un ordinador interpreta per a resoldre un problema o una funció específica."

El conjunt de programes per a fer diferents funcions d'un dispositiu electrònic s'anomena Software.

**1.2.1 Explicació del codi font:**

El codi font és una de les parts més importants d'un programa. El codi font d'un programa és el conjunt de línies que un programador escriu amb un llenguatge de programació amb els passos que ha de seguir un dispositiu per executar dit programa.

Un exemple de codi font és el de la figura 1, que com es pot apreciar és un conjunt de línies escrites en un llenguatge de programació ( en aquest cas el llenguatge utilitzat és Java). Aquestes línies de codi són essencials per a que el programa, quan sigui executat, tingui les ordres que ha de fer.



1 Exemple de codi font. Font: Imatge pròpia

**1.2.1.1 Explicació de les paraules reservades en el codi font:**

Dintre del codi font, encara que aquest depengui del llenguatge de programació, es poden trobar ítems que es trobaran en tots els llenguatges de programació. Un d'aquests ítems son les paraules reservades.

Les paraules reservades, són identificadors dintre d'un codi font, predefinits pel propi llenguatge de programació que tenen un significat "especial" assignat i que no es poden utilitzar com identificadors de qualsevol element en el seus programes.

Per a entendre millor aquest fet, s'introduirà una paraula reservada en el codi font d'un programa de java (però podria ser aplicat a qualsevol llenguatge de programació, amb qualsevol paraula reservada d'aquest.)

En java, una paraula reservada es per exemple la paraula "int". Per tant, is li assignem nosaltres un identificador, el programa ens donarà error a l'hora de executar l'acció.

Figura 2 Exemple de paraules reservades Font: Pròpia

Com es pot apreciar en la figura 2, si fiquem una la paraula int en qualsevol lloc de el programa (exceptuant algunes funcions, que seran explicades a la part pràctica) el llenguatge la detectarà com a una paraula reservada.

En canvi com la paraula "noerror" no és una paraula reservada, el codi s'executarà sense cap error.

**1.2.1.2 Explicació de les anotacions:**

En gairebé tots els llenguatges de programació, podem trobar una eina que s'utilitza per a organitzar el codi font i explicar a altres programadors per a què serveix i que fa aquell codi font. Aquesta eina són els anomenats símbols d'anotacions. En cada llenguatge s'associen dos o tres símbols que son ignorats pel programa i que l'únic que fan és que el programador que escrigui el codi pugui explicar per a què serveix qualsevol línia del codi font, i per tant, donar informació a altres programadors o usuaris que entenguin el llenguatge de programació.

En el cas de java, c o c++ les anotacions simples són donades amb els símbols "// " com es pot apreciar en la figura 3.



Figura 3 Anotacions Font: Pròpia

**1.2.2 Executable**

Un executable és el conjunt d'arxius que formen un programa. Això vol dir que un executable inclou el codi font i, a més a més, altres ítems que poden ajudar a fer el programa més extens o comprensible. Un exemple d'arxius que es poden trobar dintre d'un executable poden ser imatges o arxius en llenguatge de programació XML ( llenguatge de programació orientat a interfícies gràfiques) que no executen cap acció sinó que donen una interfície gràfica que pot ajudar a l'usuari que executi el programa (un exemple pot ser el arxiu icon.png de la figura 4)

Aquests arxius addicionals es solen ficar en carpetes per a diferenciar-ho d'altres arxius i per a tindre un executable organitzat (com es pot apreciar en la figura 4, en l'el·lipse de color vermell).



Figura 4 Exemple d'executable F ont de la figura: pròpia, codi: aliè Font de Simple-calculator: <https://github.com/SimpleMobileTools/Simple-Calculator>

**1.2.3 Definició d'aplicació**

Per a poder explicar el terme aplicació, s'ha de tornar a explicar la definició de programa per poder entendre completament certes diferencies.

Tal i com s'ha mencionat anteriorment, un programa es un conjunt d'instruccions escrites amb un llenguatge de programació que un aparell interpreta per a resoldre un problema o una funció específica.

Una aplicació no és més que un estil de programa dissenyat únicament per facilitar a l'usuari la realització d'un determinat treball. Això marca una gran diferencia amb altres programes que estan pensats per realitzar treballs més avançats i que no afecten de manera comú al usuari (com podrien ser per exemple programes que fan funcionar un sistema operatiu).

Un exemple d'aplicació pot ser un programa que fa el treball de calcular les notes finals dels treballs de recerca a partir de certs percentatges donats.

**1.3 Diferents tipus de llenguatge de programació i la seva utilitat:**

Tipus de llenguatges de programació segons la seva dificultat:

.- Llenguatges de programació de nivell baix: Es programen tenint en compte les característiques del processador al qual se li donen les ordres. Solen ser ordres lògiques i no tenen molta complexitat.

Exemples de llenguatges de programació de nivell baix: Basic, python o bash.

.- Llenguatges de programació de nivell mitjà: Són llenguatges de programació que ajuden als llenguatges de programació de nivell alt. Són molt útils implementant algoritmes, per fer taules mitjançant codi, etc.

Exemples de llenguatges de programació de nivell mitjà: C o Java-script.

.-Llenguatges de programació de nivell alt: Són els llenguatges de programació que més s'aproximen al llenguatge dels éssers humans. Això vol dir que a vegades ignoren el funcionament de la màquina. Són molt utilitzats en Intel·ligència Artificial.

Exemples de llenguatge de programació de nivell alt: Java, Ruby o Kotlin.

**1.4 Tipus de llicències en els llenguatges de programació i la importància d'aquestes:**

Per a poder explicar les llicències en els llenguatges de programació, s'ha d'entendre primer el significat de la paraula llicència. Segons l' IEC, la paraula llicència té el següent significat: "Llibertat de fer o de dir alguna cosa en virtut d’una permissió donada." Per tant, si apliquem la paraula llicència al món de la programació, trobem que el significat variarà, ja que una llicència en el món de programació i la informàtica és un contracte entre el creador d'un programa, software o un llenguatge de programació i un programador o un usuari comú. Dintre d'aquesta llicència, que ha d'existir, en podem trobar de diferents tipus, les llicències de software lliure (on el "contracte" explicat anteriorment seria el idoni per a l'usuari o programador) i les llicències de software propietari (on el " contracte explicat anteriorment, seria idoni pels creadors del llenguatge de programació).

**1.4.1 Les Llicències de Software Lliure:**

Les llicències de software lliure, tal i com s'han comentat abans, són aquelles llicències que, el "contracte entre el usuari/programador i els creadors del llenguatge de programació o software beneficien sobretot als usuaris. Aquest benefici sorgeix dels següents motius:

.- El codi del llenguatge/software és accessible per tothom.

.- No té cost per utilitzar ell llenguatge o software

.- Dona total llibertat a l'usuari.

.- No requereix actualitzacions.

.- Es pot trobar orientacions a fòrums, blogs i wikis ( que solen ser més properes als usuaris).

Un exemple de llicència de Software lliure és la llicencia anomenada GNU LGPL , què és la que utilitzen la majoria de llenguatges de programació lliures, com ara Java ( figura 5).



Figura 5 Llicència GNU i la seva col·laboració amb Java Font: Pròpia

**1.4.1 Les Llicències de Software Propietari:**

Les llicències de software propietari, són les llicències que beneficien sobretot a els creadors d'un llenguatge de programació o software. Els motius d'aquest benefici són els següents:

.- El usuari no té accés al codi.

.- Requereix actualitzacions

.- Té un determinat cost

.- Limita l'ús de l'usuari

.- Orientació únicament mitjançant pàgines oficials y manuals autoritzats.

Un exemple de software propietari pot ser Windows, que compleix tots els punts anomenats anteriormen

**2. Compiladors :**

**2.1 Definició de compilador i la seva funció**

Segons Advanced Compiler Design and Implementation de Steven S. Muchnick, " Un Compilador son sistemes de software que tradueixen programes escrits en llenguatges de programació avançats en el seu equivalent en llenguatge màquina o codi objecte per a la seva execució en determinada màquina " – definició agafada del llibre Advanced Compiler Design and Implementation, Steven S. Muchnick.

**2.2 Fases del Compilador**

Un compilador té diferents fases des de que s’executa fins que treu el producte finalitzat, ja sigui un arxiu en codi màquina o un executable. Segons el llibre Advanced Compiler Design and Implementation de Steven S. Muchnick, els compiladors segueixen sempre 4 fases:

1. Anàlisis lèxic: El compilador rep el codi font, analitza el codi font i separa les línies de codi per a transformar-les en tokens. El compilador té aquesta habilitat de fer un anàlisis lèxic del codi font ja que s’ha programat per identificar les paraules reservades del codi font, els identificadors d’aquestes, etc.

Un token, consisteix en agafar les línies de codi independents i separar-les com si fossin frases, pera després analitzar-les.

1. Anàlisis Sintàctic o anàlisi: Es processen la seqüència de tokens i es produeix un arbre d’anàlisis sintàctic ( com es faria a l’hora d’analitzar una frase ) per a comprovar si l’expressió dels tokens es sintàcticament correcta. Després d’aquest anàlisis es genera un codi entremig, que es un codi entremig de el llenguatge de programació i el codi màquina
2. Procés de comprovació de la semàntica: Es comprova si el anàlisis d’arbre construït segueix les regles del llenguatge de programació, com per exemple si els identificadors del codi font s’identifiquen abans d’utilitzar-se o no
3. La generació del codi: Transforma el codi entremig en el seu equivalent en codi màquina.

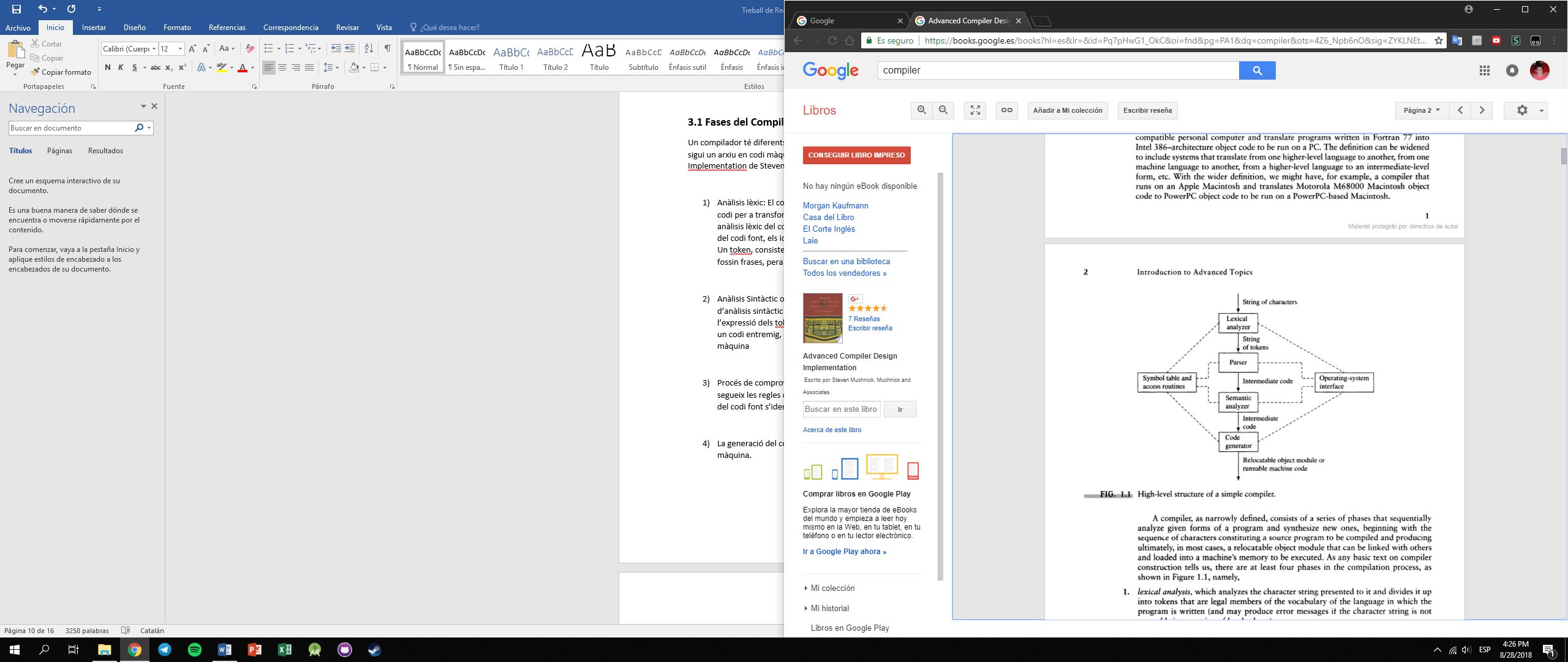


Figura 6 Fases del Compilador Font: Advanced Compiler Design and Implementation

**2.3 Tipus de Compiladors**

Es poden trobar diferents tipus de compiladors segons el tipus de procediment que utilitzin per a aconseguir convertir el codi en un llenguatge entès per una màquina. En aquest cas, es mostraran el tipus de compiladors que són més útils en Java i en Android:

Compiladors JIT:

Compiladors AOT:

Compiladors ByteCode:

**3. Introducció a Java:**

**3.1 Definició i característiques del llenguatge de programació Java:**

Java és un llenguatge de programació multi plataforma, amb un propòsit general orientat a objectes.

En una única frase, s'han pogut introduir tres nous conceptes: el llenguatge de programació multi plataforma , un propòsit general i la programació orientada a objectes. Ara es definiran breument aquests conceptes que després s'explicaran més detalladament.

.- El llenguatge de programació multi plataforma: És aquell llenguatge de programació on el seu codi pot ser executat en qualsevol màquina mitjançant un compilador.

.-Llenguatge de propòsit general: Són aquells llenguatges de programació que poden ser utilitzats per a diferents propòsits, com podrien ser des de la creació d'aplicacions o les bases de dades fins a càlculs matemàtics.

.-Programació orientada a objectes: Es una nova filosofia de programar, que canvia el tipus de programació lògica de les màquines a una altra més adaptada al pensament humà.

**3.1.2 Programació orientada a objectes:**

Segons Elivar Largo ( desenvolupador de Software durant 4 anys) explica la programació orientada a objectes com la forma d’agrupar característiques similars de diferents objectes anomenades classes, i a partir d’aquestes classes fer que aquests objectes es comuniquin entre ells mateixos mitjançant missatges de codi, per tal de que la unió d’aquests missatges i dels objectes de les determinades classes formin un programa.

A partir d’aquesta definició, han sorgit dos tipus de paraules noves molt importants que defineixen la programació dedicada a objectes:

1. Classe: Una classe es una plantilla, un motlle o un model per a aconseguir crear objectes. Una classe està composta per unes propietats i atributs ( anomenats variables ) i per uns comportaments ( anomenats mètodes) que creen un tipus de model a seguir que després seguirà un objecte. Un exemple de classe podria ser una bicicleta. On les seves variables serien que té dues rodes, un manillar i uns frens i els mètodes serien que pot accelerar i frenar.
2. Objecte: Un objecte es una instància d’una classe, un cas específic d’aquesta. Per a entendre-ho millor, Elivar ens cita un exemple de com seria una classe de mòbils amb diferents objectes dintre d’aquesta. El exemple consta de que encara que la classe sigui definida per a mòbils no tots els mòbils son iguals , tenen diferent tamany, pes, color, etc. Però segueixen compartint unes característiques bàsiques que fan que aquells objectes siguin mòbils, com podrien ser la funcionalitat de trucar o de fer fotografies.

**3.2 Introducció a l’ecosistema de java.**

Java és un llenguatge de programació multi plataforma, el que vol dir que funciona en tots els dispositius, però per a que el codi font pugui funcionar en tots els sistemes operatius i tots els processadors disponibles en el mercat, les aplicacions que són escrites per aquest llenguatge de programació, necessiten un ecosistema on puguin convertir aquest codi, en un executable compatible per a un sistema operatiu ( en el cas de Windows un arxiu .exe, en cas de mac os .app)

Aquest ecosistema d’aplicacions que convertiran el codi font en un executable per al nostre sistema operatiu venen en una aplicació anomenada java JDK.

**3.2.1 Java JDK:**

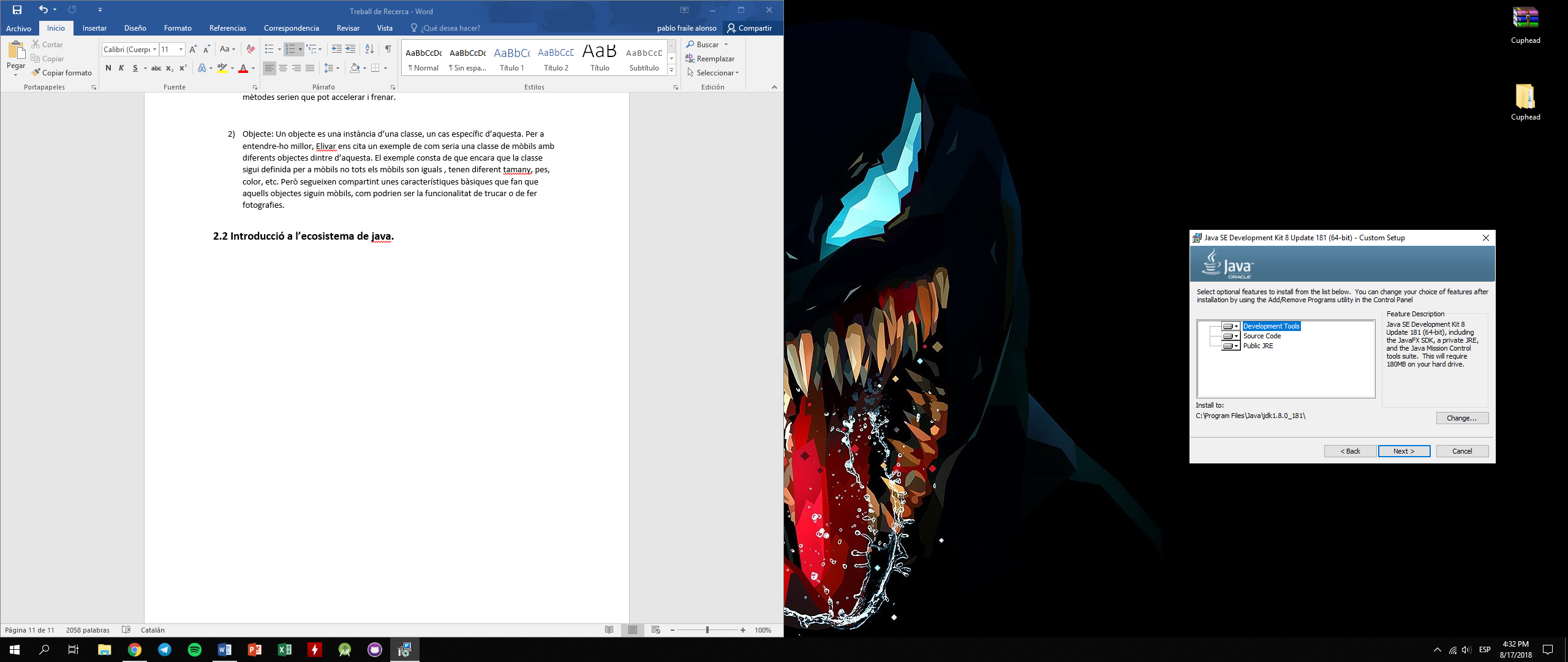
Java JDK (Java Development Kit) és un instal·lador de tots els diferents programes necessaris per a poder programar i executar un codi font java en qualsevol sistema operatiu.

Aquests programes indispensables per a crear un executable programat en java i que el JDK inclou són els següents:

.- JavaFX ( SDK ): Proporciona diferents ordres i tecnologies per a poder desenvolupar aplicacions relacionades amb navegadors, ordinadors o dispositius mòbils.

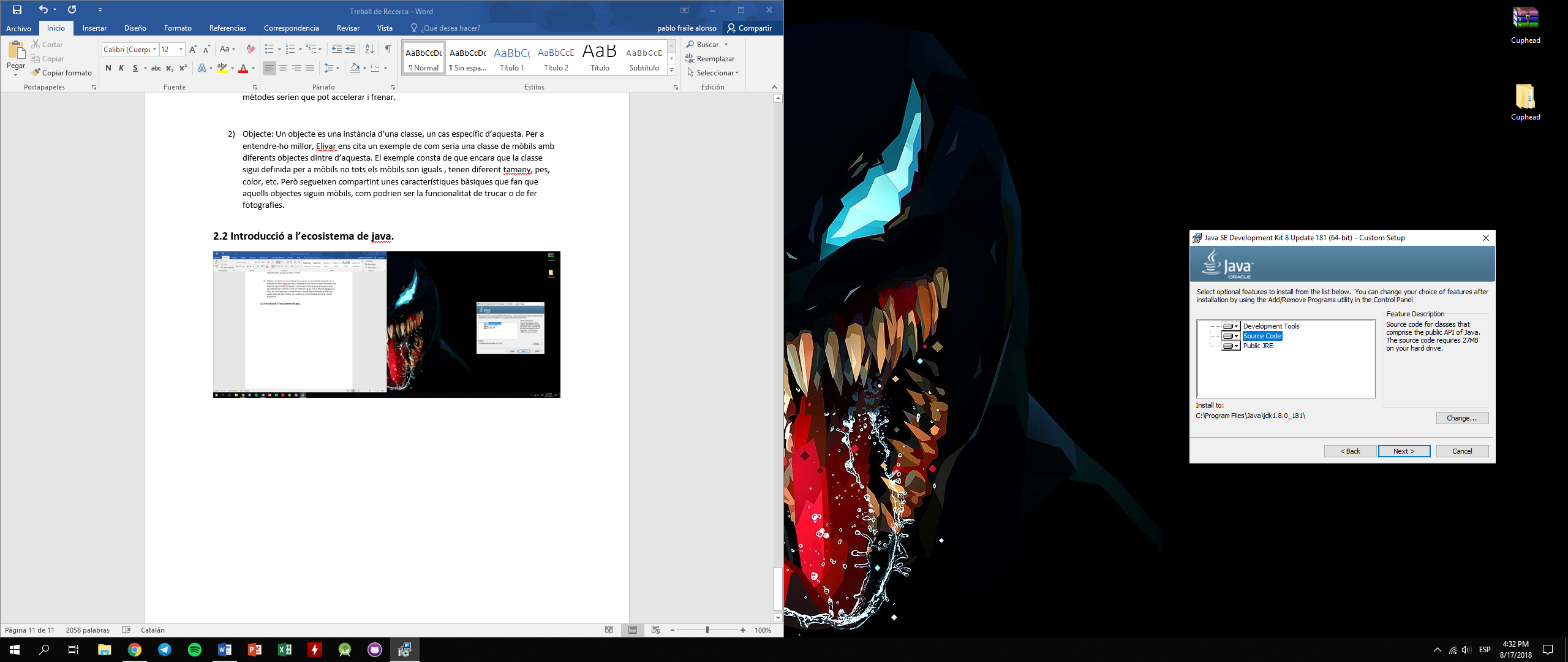
.- Java Mission Control: Eina que proporciona tot tipus d’informació (col·lecta d’errors i funcionament de l’executable o la aplicació ) al desenvolupador.

.- Private Java JRE : Es utilitzat per a la compilació d’una aplicació Java feta per un usuari.



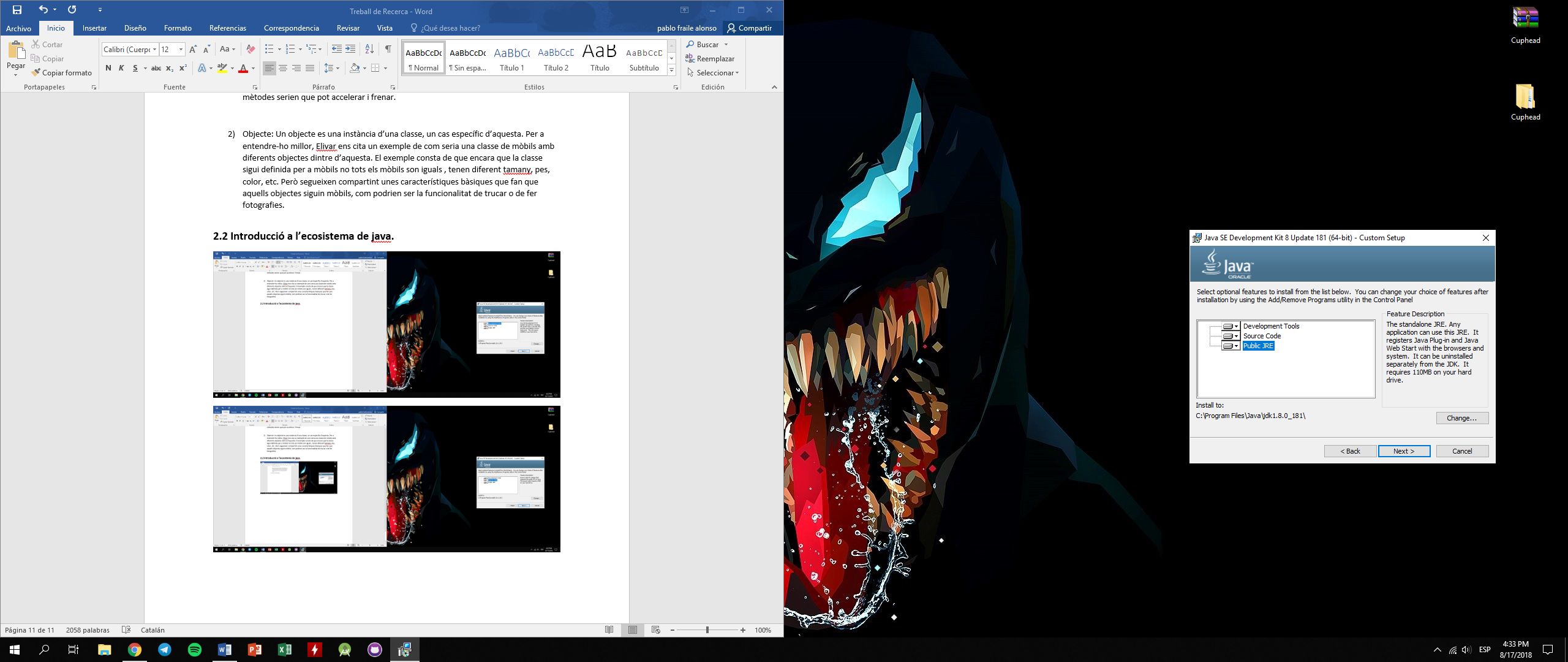
Instalació de java JDK ( Windows) Font: Pròpia

.- Source Code : Instal·la l’API ( Application Programming Interface, o en Català, Interfície de programació d’aplicacions) necessària per a poder treballar amb totes les opcions que els desenvolupadors de el llenguatge de programació Java han preparat per a l’escriptura del codi font.



Instalació de java JDK ( Windows) Font: Pròpia

.- Public Java JRE: Es utilitzat per totes les aplicacions java (no únicament per compilar les del usuari) a més a més d’incloure un plug-in per al navegador d’internet en cas de que una pàgina tingui que fer anar un codi java o java-script (variant de java per a pàgines html, normalment pàgines web.



Instalació de java JDK ( Windows) Font: Pròpia

**3.3 Creació de l’aplicació a partir del codi font**

Una vegada explicat totes les aplicacions necessàries per a executar una aplicació java i després d’entendre què és el codi font, cal explicar com es passa de ser unes lletres de codi, a ser un executable totalment funcional en qualsevol sistema operatiu.

En el cas d’aquest llenguatge de programació el que fa que sigui compatible amb tots els dispositius no es el codi en si, si no l’aplicació JRE que inclou el paquet d’aplicacions de java JDK.

Aquesta aplicació és única de cada sistema operatiu, i de moment està disponible per a Windows, Mac OS i Linux. Això vol dir que si java JRE (o una variant d’aquesta, com per exemple Dalvik o ART, en el sistema operatiu Android) no és compatible amb un sistema operatiu, aquest no podrà executar cap aplicació programada amb java, al menys fins a què s’adapti una versió de la Java Virtual Machine al sistema.

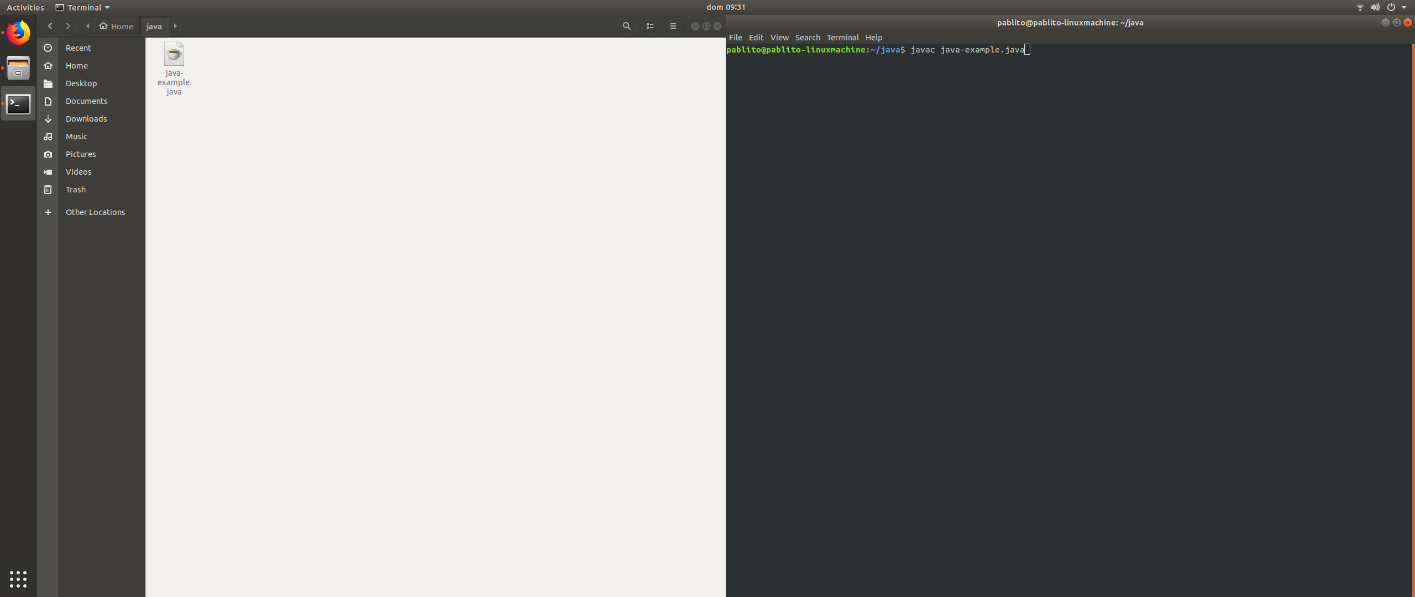
**3.3.1 Procés de compilació**

La compilació és el procés de convertir les diferents classes definides en el codi font escrites en el llenguatge de programació java en un arxiu en format ".class". Els arxius class contenen dintre un codi binari especialitzat per a la Java Virtual Machine anomenat ByteCode. Com el codi binari ByteCode es llegible per la Java Virtual Machine, aquest, continua sent multiplataforma en tots aquells sistemes que suportin la JVM.

El compilador que s’utilitza per a passar d’un arxiu .java a un arxiu .class s’anomena javac ( de les sigles java compiler). Aquest programa s’ha d’executar una finestra de comandaments, tant a Windows, Mac o Linux.

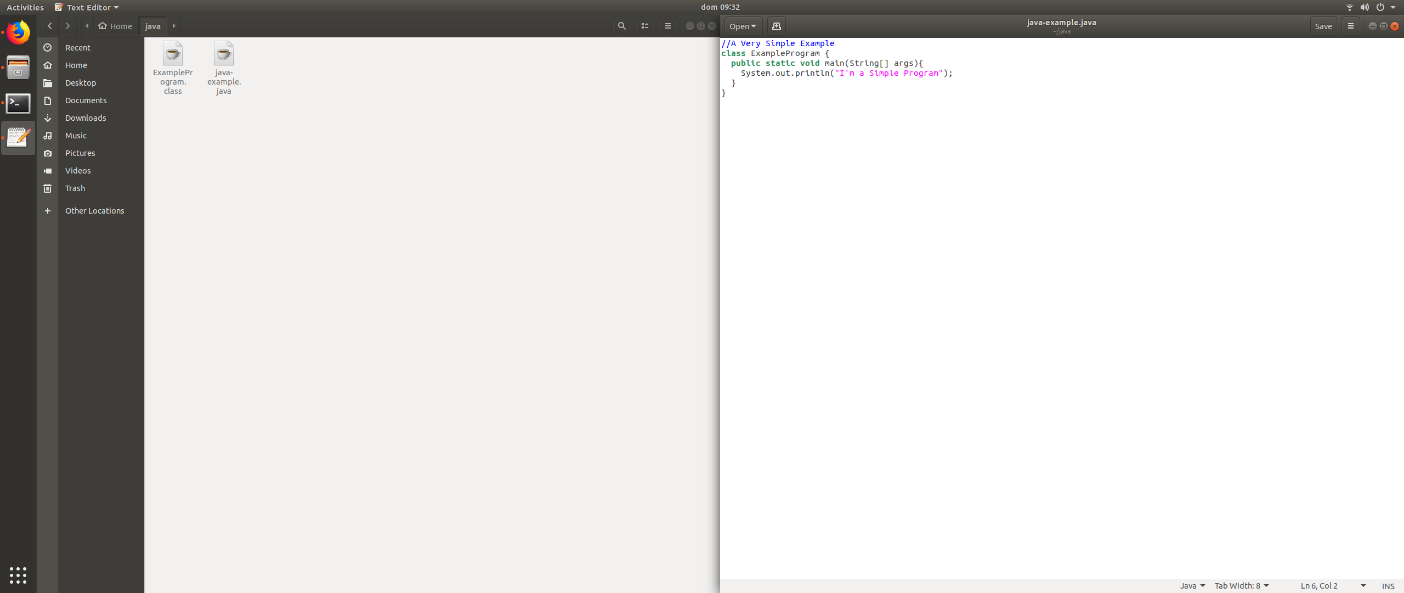
En aquest cas, per a comprovar com funciona el procés de compilació java, s’ha utilitzat un sistema operatiu Linux ( ja que sol ser més fàcil en quant a tema de comandaments i compilació en diferents llenguatges de programació). El comandament seria el mateix en Mac OS i en Windows.

Com es pot apreciar en la figura 2.3.1.1, tenim un arxiu anomenat java-example.java, i una finestra de comandaments on tenim el nom del compilador (javac) i el arxiu que volem compilar.



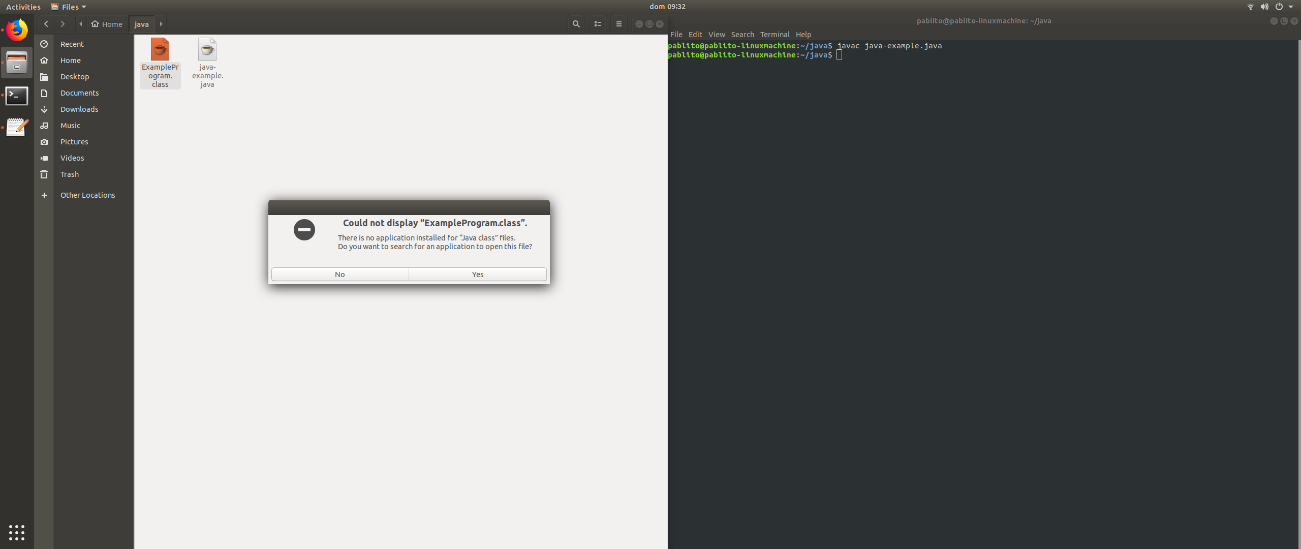
2.3.1.1 Comandament Javac Font de la imatge: pròpia, codi: Oracle, Java

Una vegada executem el comandament, es pot apreciar a la figura 2.3.1.2 que automàticament se’ns crea un arxiu amb el nom de la classe programada en el arxiu java-example.java . Això és perquè el compilador crearà tants arxius en bytecode com classes hi hagin al arxiu .java (ja que les classes son les encarregades de donar ordres als programes) .



2.3.1.2 Creació arxiu .class Font de la figura: pròpia, codi: Oracle, Java

En el cas de que es vulgui obrir un arxiu .class (figura 2.3.1.3) , el Sistema no serà capaç de llegir-lo, ja que és un idioma màquina que només la Java Virtual Machine interpreta.



2.3.1.2 Intent d’obrir un arxiu .class Font de la figura: pròpia, codi: Oracle, Java

Quan es té múltiples classes, i per tant, múltiples arxius .class, es comprimeixen tots en un arxiu .jar, que prové de les sigles Java ARchive (que en català vol dir arxiu de java) que és una variació d’un arxiu .zip. El arxiu .jar, és igualment llegit per la Java Virtual Machine.

**3.4 Java API**

La paraula API prové de les sigles Application Programming Interface, que en català significa Interfície de Programació d’Aplicacions. En el cas de Java, una API es una llista de totes les classes predefinides que són part del Java Development Kit. L’API inclou tots els paquets de Java, classes, interfícies conjunt amb els seus mètodes i constructors.

Totes aquestes classes pre-escrites que són guardades en l’API de java són de gran ajuda per a els programadors, i aquests tenen que estar atents a les diferents actualitzacions o modificacions d’aquestes. Totes les diferents classes que venen instal·lades amb el Java JDK es poden trobar en el següent document de Oracle: <https://docs.oracle.com/javase/7/docs/api/>

En cas de que algun programador vulgui utilitzar una classe de la Java API, necessita incloure una declaració de que està important aquesta classe al seu codi al principi del seu codi font. Per exemple, si es vol utilitzar la classe “ Scanner” que dona permís al programa per a acceptar text escrit per el usuari amb el teclat, es tindria que incloure la següent línia de codi al codi font:

import java.util.Scanner;

Aquesta declaració d’importació de la classe permet que el programador pugui utilitzar qualsevol mètode inclòs en la classe Scanner.

En cas de el java JDK, sol incloure en tots els arxius del codi font, diferents declaracions d’importació que s’utilitzen molt sovint, com per exemple java.util o java.lang (que incorporen les classes de les variables, classes de suma,resta,etc).

**4. Android Studio :**

**4.1 Introducció a Android Studio.**

**Android Studio:**

Android Studio és el IDE oficial per a la creació d’aplicacions Android. Aquest IDE (Integrated Development Environment, també anomenat ambient integrat per a desenvolupar) està basat en el projecte IntelliJ IDEA, un dels projectes IDE més grans per a diferents llenuatges de programació i en sobretot Kotlin (un llenguatge de programació basat en Java).

Android Studio, incorpora diferents característiques i utilitats per a la creació d’aplicacions com per exemple:

.- Un sistema de construcció amb la eïna Gradle

.- Un emulador Android per a poder provar la app sense un dispositiu

.- Un ambient unificat on es pot desenvolupar per a tots els dispositius android

.- Opció d’aplicar canvis a la app sense que tindre que compilar tota la apk de nou

.- Plantilles de codi i la integració de la pàgina web github per a poder importar diferents parts de codi d’altres usuaris

.- Una varietat d’eïnes per a testejar l’app i els frameworks

.- Eïnes per a poder testejar l’usabilitat, la compatibilitat i la potència de l’aplicació.

.- Compatibililtat amb altres llenguatges de programació com C++ o NDK

.- Integració amb la plataforma de google cloud, fent més senzill d’integrar servicis de google com per exemple el Google Cloud Messaging

**Estructura d’un projecte en Android Studio:**

Cada projecte d’Android Studio conté un o més mòduls de codi font i arxius de Recursos. Els arxius de Recursos són tots aquells arxius addicionals i contingut estàtic que el codi font utilitza, com per exemple tota la part gràfica ( animacions, interfície, etc.)

Els diferents tipus de mòduls que podem trobar inclouen:

· Mòduls de l’Android App

· Mòduls de Llibreries.

· Mòduls del Google App Engine

De base, Android Studio mostra els arxius del projecte en l’apartat d’Android Studio anomenat la vista del projecte Android ( figura 4.1.1). Aquesta vista organitza els arxius per mòduls per a proporcionar una vista més ràpida i ordenada.

Tots els arxius de compilació són visibles davall de Gradle Scripts, mentre que a la part superior es pot trobar els diferents mòduls de la app organitzats en les següents carpetes:

· manifests: Conté el arxiu AndroidManifest.xml

· java: Conté el o els codis fonts programats en java

· res: Inclou tots els recursos que no varien, com per exemple la interfície d’usuari, imatges,etc.

La vista de projecte Android pot variar depenent de la opció que se li encomana, per exemple, com es pot apreciar a la figura 4.1.1, per a veure la estructura del projecte, estem utilitzant la vista Android però també es poden utilitzar diferentes vistes depenent de la situació. Per exemple, si es selecciona la vista “Problems” ( figura 4.1.2) es mostren tots els arxius que contenen algun error sintàctic o elements no declarats.



**Interfície d’usuari en Android Studio:**

La finestra principal d’Android Studio està composta de diferents àrees, com es pot apreciar en la figura 4.1.3.



Figura 4.1.3 Font de la figura: Android Developers

1. La barra d’eines proporciona diferents accions, com per exemple executar l’aplicació des de l’emulador inclòs, o simplement compilar l’aplicació.
2. La barra de navegació, ajuda a l’hora de navegar per els diferents arxius dintre del propi projecte. Incorpora una vista reduïda de la finestra de la vista del projecte.
3. La finestra d’edició: És on es pot modificar i crear el diferent codi. Depenent del tipus de codi modificat, la finestra pot canviar. Per exemple si s’està modificant una part gràfica de l’aplicació, la finestra serà diferent que si s’està programant el codi font.
4. La barra de finestres d’eines, es troba al voltant d’Android Studio i conté els botons que permeten expandir o minimitzar diferents finestres amb eines.
5. La eina de finestres, dona accés específic a tasques com la gestió de projectes, búsqueda, versió de l’aplicació, etc.
6. La barra d’estat treu per pantalla els estat del projecte i de el IDE, així com també diferents avisos o missatges.

**4.2 Android SDK**

El Android SDK (Software Development Kit, en català kit de desenvolupament de programació) és una composició de múltiples aplicacions que són requerides per al desenvolupament de la plataforma Android.

Dintre del Android SDK, es pot trobar que depenent de la versió d’Android per a la que vulguem programar una aplicació, tindrem que baixar una versió del SDK o un altra, ja que cada versió del SDK inclou les diferents API per a poder programar l’app, i si es programa una aplicació amb una API més antiga de la que pertoca al dispositiu, no es podran afegir a l’aplicació totes les funcions que la versió d’Android incorpora.

**5. Java adaptat a Android:**

**5.1 Funcionament de Android**

Segons Android Developers ( pàgina oficial de desenvolupadors d’Android), Android es un conjunt de software de codi lliure basat en Linux per a una varietat amplia de dispositius i factors de forma. En la figura 5.1.1 es mostren tots els components principals de la plataforma Android.

 Figura 5.1.1 Font: Android Developers

**Kernel de Linux:**

La base de la plataforma Android es el kernel de Linux. El kernel es la part del telèfon on es mantenen tots els controladors de el hardware que un dispositiu té incorporat, fent així que siguin llegits per el HAL . A més a més, ART la màquina virtual d’Android, també es basa en el kernel de Linux per a funcionalitats subjacents com la generació de subprocessos i la administració de memòria de baix nivell.

A més a més, el ús del kernel de Linux permeteix que Android aprofiti funcions de seguretat com per exemple IPC ( comunicació segura entre processos ) que facilita la comunicació segura entre aplicacions que funcionen en diferents processos.

**Capa d’abstracció de Hardware ( HAL ) :**

La capa d’abstracció de hardware (HAL) brinda interfícies estàndards que exposen les capacitats de hardware del dispositiu al framework de la Java API de nivell més alt. La HAL consisteix en diferents mòduls de biblioteca on cada un d’aquests incorpora una interfície per a un component de hardware específic, com per exemple el mòdul de càmera o de Bluetooth. Quant el framework d’una API realitza una trucada per a accedir al hardware del dispositiu, el sistema Android carga el mòdul de biblioteca per a el component de hardware en qüestió.

**Temps d’execució d’Android :**

Per als dispositius amb Android 5.0 ( Nivell d’API 21) o versions posteriors, cada aplicació executa els seus propis processos amb ART. El ART està programat per executar diferents màquines virtuals en dispositius de memòria baixa executant arxius en format DEX, un format de codi de bytes (com seria bytecode ) però dissenyat especialment per a Android i optimitzat per a ocupar un espai de memòria mínim. Els arxius DEX, són els que es poden executar en la plataforma Android.

Algunes de les funcions principals del ART són les següents:

· Tipus de compilació ahead-of-time (AOT) i just-in-time (JIT)

· Compatibilitat amb la depuració d’errors, informes de errors i la capacitat d’establir punts de controls per a controlar determinades parts del sistema.

Abans d’Android 5.0 ( API 21 ), Dalvik era el temps d’execució del sistema operatiu. Si una aplicació s’executa bé en ART, també tindria que funcionar bé en Dalvik, encara que es possible que no succeeixi el contrari

**Biblioteques C/C++ natives**

Molts dels components i serveis centrals del sistema Android, tal com ART i HAL, es basen en codi natiu que requereix les biblioteques natives escrites en C i C++. La plataforma Android proporciona la API del framework Java per a exposar la funcionalitat d’algunes d’aquestes biblioteques natives a les aplicacions. Per exemple, es pot accedir a OpenGL ES ( llibreria per a gràfics 2D i 3D ) a través de la Java OpenGL API que incorpora el framework d’Android per a que les aplicacions puguin agreagar compatibilitat amb dibuixos i gràfics.

**Framework de la Java API:**

Tot el conjunt de funcions del Sistema Operatiu (S.O) està disponible mitjançant APIS escrites en el llenguatge Java. Aquestes APIS son els ciments que es necessiten per a crear aplicacions android simplificant la reutilització de components del sistema i serveis centrals i modulars, com les següents:

· Un sistema de vista: Enriquidor i extensible que es pot utilitzar per a compilar la interfície d’usuari (IU) d’una aplicació. Aquesta inclou llistes, quadrícules, quadrats de text, botons e inclòs un navegador web integrable. Aquest sistema de vista es pot incorporar a la aplicació amb els objectes View i ViewGroup.

· Un administrador de recursos que brinda accés a els recursos (res) de l’aplicació, com per arxius de disseny.

· Un administrador de notificacions que permet que totes les aplicacions mostrin alertes personalitzades en la barra de notificacions del Sistema Operatiu.

· Un administrador d’activitat que administra el cicle de vida de les aplicacions i proporciona una pila d’activitats comuna.

· Proveïdors de contingut que permeten que les aplicacions accedeixin a dades des de altres aplicacions com per exemple que una aplicació agafi dades de l’aplicació de contactes o comparteixin les seves pròpies dades.

Els desenvolupadors tenen accés total a les mateixes APIS del framework que utilitzen les aplicacions del sistema Android. Tots els paquets d’APIS proporcionats per els desenvolupadors d’Android es poden trobar aquí: <https://developer.android.com/reference/packages?hl=es-419>

**Aplicacions del sistema :**

Android inclou un conjunt d’aplicacions centrals per a correu electrònic, missatgeria SMS, calendari, navegació en internet, contactes, entre altres elements. Les aplicacions incloses en la plataforma no tenen un estat especial entre les aplicacions que el usuari decideix instal·lar. Per això, una aplicació externa es pot convertir en el navegador web, el sistema de missatgeria o inclòs en el teclat predeterminat per l’usuari ( encara que existeixen algunes excepcions com per exemple l’aplicació configuració del sistema)

Les aplicacions del sistema funcionen com aplicacions per a usuaris i brinden capacitats claus a les quals els desenvolupadors poden accedir des de les seves propis aplicacions. Per exemple, si una aplicació intenta integrar un missatge SMS, no es necessari compilar aquesta funcionalitat un altre cop; com alternativa es pot invocar la aplicació de SMS que ja està instal·lada per entregar un missatge al receptor especificat.

**5.2 Funcionament del Android Runtime (ART)**

Android Runtime ( ART ) és la màquina virtual utilitzada per les aplicacions i alguns servicis del sistema en el sistema operatiu Android. ART i el seu predecessor Dalvik van ser originalment creats especialment per el projecte Android. ART pot executar els formats Dalvik Executable i DEX bytecode.

ART i Dalvik tenen compatibilitat executant arxius en format DEX Bytecode, així que aplicacions desenvolupades per a l’arquitectura de màquina virtual Dalvik, tindrien que funcionar amb ART. Malgrat això, algunes tècniques que funcionen amb Dalvik, no tenen perquè funcionar amb ART

Millores i afegits de l’arquitectura de la Màquina Virtual ART respecte Dalvik:

**· Mètode de Compilació:**

ART introdueix el tipus de compilació anomenat ahead-of-time (AOT), vist anteriorment en el apartat 2 d’aquest TdR.

Aquest nou tipus de compilació, incrementa el rendiment de les aplicacions respecte el mètode Just In Time ( JIT) utilitzat per Dalvik.

<https://source.android.com/devices/tech/dalvik/>

ART 🡪 AOT ( Ahead Of Time)

Dalvik 🡪 JIT ( Just in Time)