BLOC A

Jaume Minerva Tomas
1r A
Tortosa, 7 d'octubre del 2021

ÍNDEX

BLOC A **1**

1. Entorn integrat de desenvolupament **3**

2. Comentaris al codi **4**

3. Fonaments de programació **5**-7

4. Disseny d'algorismes. **8-9**

5. Prova de programes.Depuració d'errors **10**

6. Documentació dels programes **11**

7. Entorn de desenvolupament de programes **12**-14

1. Entorn integrat de desenvolupament

Un entorn integrat de desenvolupament o IDE , és una eina informàtica per al desenvolupament de programari de manera còmoda i ràpida. Així doncs és un entorn de desenvolupament que agrupa diferents funcions en un sol programa, habitualment: editor de codi,compilador, depurador i un programa de disseny d'interfície gràfica.
Els IDE estan dissenyats per maximitzar la productivitat del programador proporcionant components molt units amb interfícies d'usuari similars. Els IDE presenten un únic programa en què es porta a terme tot el desenvolupament. Generalment, aquest programa sol oferir moltes característiques per a la creació, modificació, compilació, implementació i depuració de programari. Això contrasta amb el desenvolupament de programari utilitzant eines no relacionades, com ara l'editor Vi, GNU (GCC) o Make.
Un dels propòsits dels IDE és reduir la configuració necessària per reconstruir múltiples utilitats de desenvolupament, en comptes de proveir el mateix set de serveis com una unitat cohesiva. Reduint aquest temps d'ajustos, es pot incrementar la productivitat de desenvolupament, en casos on aprendre a fer servir un IDE és més ràpid que integrar manualment totes les eines per separat.
Una millor integració de tots els processos de desenvolupament fa possible millorar la productivitat en general, més que únicament ajudant amb els paràmetres de configuració. Per exemple, el codi pot ser contínuament armat, mentre és editat, preveient retroalimentació instantània, com quan hi ha errors de sintaxi. Això pot ajudar a aprendre un nou llenguatge de programació d'una manera més ràpida, així com les seves llibreries associades.
El principal avantatge és que facilita la tasca del programador, mentre que l'inconvenient més important és que pot provocar mals hàbits a l'hora de programar o provocar errors que a priori, començant de zero, no es produirien.

2. Comentaris al codi

Els comentaris són anotacions que el programador incorpora al seu codi font per fer-lo més comprensible. Aquestes línies són ignorades per l'ordinador quan s'executa el programa, pel que són innòcues per a l'obtenció de el resultat final.
Cada llenguatge de programació estableix una simbologia per marcar els comentaris dins de el codi font. Per exemple, en pseudocodi sol usar-se una doble línia diagonal, és a dir, la barra de dividir dues vegades seguides (una cosa així //). Quan es vol escriure un comentari, es començarà escrivint les dues barres seguides de l'comentari, l'ordinador sabrà que el que apareix a la dreta de les dues barres és un comentari i ho ignorarà en l'execució. També aquesta simbologia és usada en alguns llenguatges de programació, encara que hi ha més variants.

En la majoria de llenguatges ha dues possibilitats per a això:

A. Es pot escriure una única línia de comentari (tota la línia és un comentari), o afegir el comentari a el final d'una mateixa línia de codi font. En aquests casos el comentari comença després d'un caràcter o conjunt de caràcters que el llenguatge especifica.
B. Es pot escriure un conjunt de línies de comentaris consecutives. Per a això hi haurà un caràcter (o més) per indicar on comencen les línies de comentaris i un altre caràcter (o més) per indicar on acaben. Tot el que estigui entre el caràcter (o els caràcters) de començament i el de cap és pres com a comentari i ignorat a l'executar el programa.

```
Algoritmo Programa_con_comentarios
    // Programa que pide al usuario números por teclado y dice si cada número
    // es par o impar. En el momento que el usuario teclea un número negativo
    // el programa termina.

    // Declaramos una variable llamada num de tipo entero, que será la variable
    // que almacenará los números que el usuario vaya introduciendo por teclado.
    definir num Como Entero
    // Asignamos un cero a la variable num para que tenga un valor mayor o igual
    // que cero y el programa comience a pedir números por teclado.
    num :=0;
    mientras num>=0    // Mientras el número sea mayor o igual que 0 pedimos números
        escribir "Teclee un número: ";
        // pedimos el número por teclado
        leer num;
        // Verificamos si el número tecleado es correcto para calcular si es
        // par o impar. Si el número es mayor o igual que cero procedemos a calcular
        si num>=0 entonces
            // Para calcular la paridad dividimos el número tecleado entre 2 y
            // almacenamos el resultado en una variable llamada dos. Este resultado
            // puede tener decimales si el número no es par.
            dos=num/2
            // Redondeamos el resultado de dividir por dos.
            enterodedos=trunc(dos)
            // Multiplicamos por dos el número truncado con lo que, si el número
            // tecleado era par, esta operación nos tiene que dar el mismo número.
            espar=enterodedos*2
            // Comprobamos si efectivamente el número calculado y el tecleado son
            // el mismo número o no.
            si espar=num     // Si son iguales es que el número tecleado era par.
                Escribir "El número ", num;
                Escribir "es par"
            Sino     // Si no son iguales es que el número tecleado era impar.
                Escribir "El número ", num;
                Escribir "es impar"
            FinSi
        finsi
    FinMientras
    // El usuario ha tecleado un número negativo, por tnto nos despedimos.
    Escribir "Adiós."
FinAlgoritmo
```

3. Fonaments de programació

Objectius i competències
Els objectius de l'assignatura són:

1. Conèixer i assimilar els conceptes fonamentals de l'algorísmia mitjançant l'aprenentatge i comprensió de la sintaxi i semàntica d'una notació algorísmica.
2. Conèixer i assimilar els conceptes, els mètodes i les tècniques per a poder especificar a partir de l'enunciat d'un problema el comportament precís que haurà de tenir la solució, dissenyar-la i implementar-ne el programa corresponent.
3. Entendre l'aplicació d'esquemes com una tècnica eficaç per a construir algorismes, i alhora el disseny descendent com una forma idònia d'afrontar problemes complexos en dividir-los en un conjunt de subproblemes més senzills.
4. Adquirir pràctica en l'aplicació dels conceptes anteriors en un entorn real de desenvolupament de programes.

Les competències de l'assignatura són:

1. Capacitat de dissenyar i construir aplicacions informàtiques mitjançant tècniques de desenvolupament, integració i reutilització.
2. Coneixements bàsics sobre l'ús i la programació dels ordinadors, sistemes operatius, bases de dades i programes informàtics amb aplicació a l'enginyeria.

Continguts

- Introducció a la programació

Computador, programa, algoritme, error de programació (bug), llenguatge de programació, portabilitat, intèrpret, shell, script, debugació, errors sintaxi, errors execució, errors semàntics, valor, variable, tipus, assignació, entrada, sortida, lectura, escriptura, sentència, mot reservat, expressió, operador, operand, precedència, avaluació, composició de sentències, funció, capçalera, cos, crida, paràmetres, valor de retorn, localitat.

- Introducció a l'algorísmica

Introducció a l'algorísmica. Representació de dades. Estructures de control del flux. Subalgorismes: procediments i funcions. Tipus de dades compostos. Apuntadors. Anàlisi d'algorismes. Algorismes de cerca i ordenació. Recursivitat. Tipus de dades abstractes: llistes lineals, arbres, grafs. Fitxers. Introducció al paradigma de la programació orientada a l'objecte: classes i objectes, propietats; modelat d'un problema de POO.

- Tractament seqüencial

Una seqüència és un conjunt de dades de qualsevol tipus que es caracteritzen per estar disposades «en forma de fila», ja sigui des d'un punt de vista físic o conceptual. Moltes de

les dades que manipula un programa tenen estructura de seqüència. Per exemple, un string com ara "Hola nois", pot ser considerat com una seqüència de caracters ('H', 'o', 'l',...) o una matriu de reals com una seqüència de files. Més formalment, una seqüència és una conjunt de dades tal que:

- És finit.
- Conté un element que és el primer de la seqüència
- Es pot definir la relació de "següent", que donat un element de la seqüència, diu quin ve a darrera.

Per exemple, el enters de 1 a 10 tenen estructura de seqüència atès que són un conjunt finit, tenen un primer ben definit (l'1) i la relació de següent consisteix a sumar una unitat a l'anterior.

```
l = ['hola', 'ale', 'ea', 'citron']
i = 0
n = 0
while i < len(l):
    if len(l[i]) > 3:
        n += 1
    i += 1
```

- Tipus estructurats de dades

Registre (estructura de dades), en programació, és un tipus de dada estructurada format per la unió de diversos elements sota una mateixa estructura. Aquests elements poden ser, o bé dades elementals (enter, real, caràcter,...), o bé altres estructures de dades.
A cadascun d'eixos elements se li diu camp.
Un registre es diferencia d'un vector que aquest és una col·lecció de dades iguals, és a dir, tots del mateix tipus, mentre que en una estructura els elements que la componen, encara que podrien ser-lo, no té per què ser del mateix tipus.

Exemple: Creació d'un registre (o estructura) en C

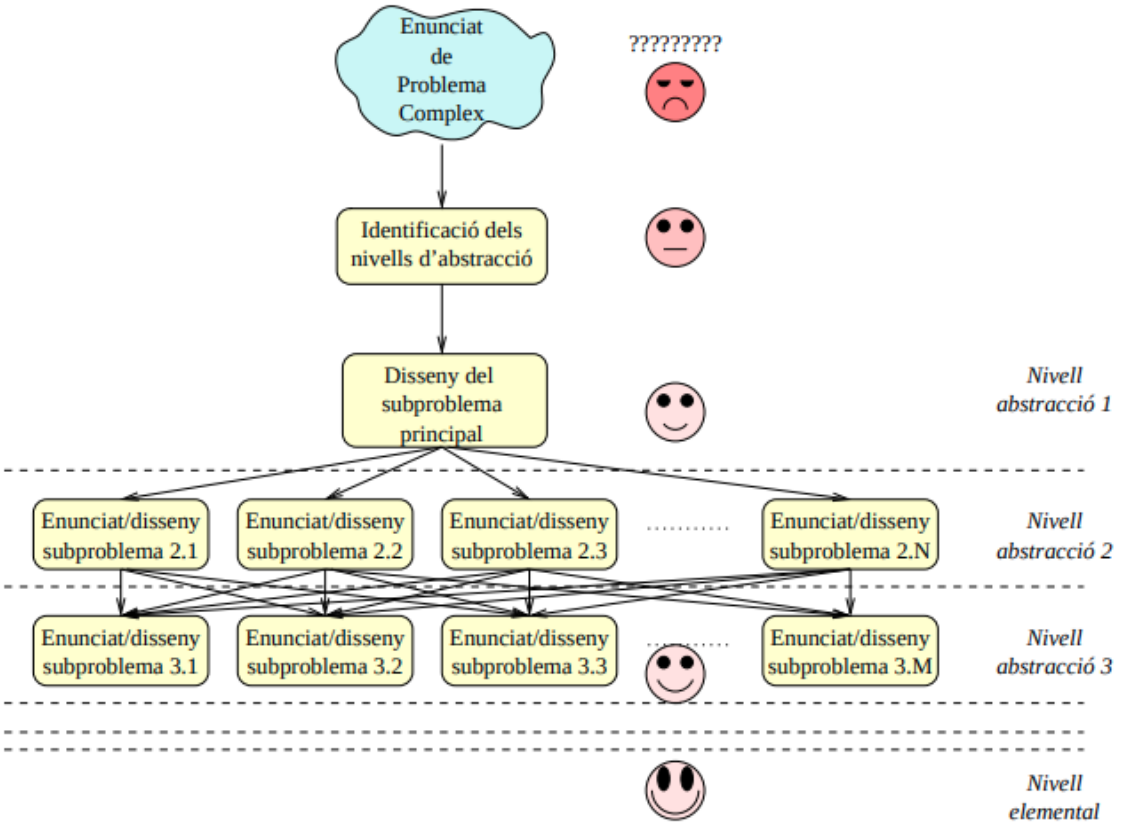
Un exemple de com es declararia un registre en C podria ser:

```
typedef struct TipoNodo
{
int dato;
struct TipoNodo *sig;
struct TipoNodo *ant;
} TNodo;
```

En aquest exemple es defineix el tipus de dada TNodo (o struct TipoNodo, seria equivalent) com una estructura (registre) que conté una dada de tipus enter i dos punters sig i ant (següent i anterior) que serveixen per a referenciar a altres registres del tipus TNodo.
Aquesta és l'estructura de dades que se sol utilitzar com node en les llistes doblement enllaçades.

- Introducció a la metodologia de disseny descendent

El disseny descendent consisteix en descomposar un problema complex en problemes més senzills.La paraula "descendent" indica que la descomposició calfer la des del nivell més abstracte del problema al més concret i no al revés.



No tota descomposició d'un problema en subproblemes és adient per obtenir un bon disseny.
La bondat de la descomposició rau en les abstraccions que convenia adequades pel problema que analitzem i que porten a una descomposició en subproblemes que són molt independents entre sí.
Quan decidim les abstraccions a fer, les podem orientar a:
Objectes del problema no elementals.Definirem un tipus apropiat a l'objecte i associarem mitjançant accions i/o funcions, les operacions que poden tenir. '
Tractament del problema (funcional). El Tractament del problema conve formularlo en varies 'etapes funcionals.

4. Disseny d'algorismes.

Un algorisme informàtic és un procés seqüencial d'instruccions finites amb l'objectiu de prendre una decisió o bé trobar la solució a un problema específic. Una sèrie complexa

d'algorismes informàtics ordenats i codificats mitjançant el llenguatge de la programació, esdevé un programa.

Partint de la base que els alumnes tenen uns coneixements bàsics sobre programació i estructures de dades, es pretén que l'alumne sigui capaç de analitza, dissenyar i implementar algoritmes basant-se en les tècniques de disseny d'algoritmes existents. Per complir amb aquest objectiu, l'alumne adquirirà els coneixements sobre: Especificació formal de problemes. Com passar d'una descripció d'un problema a una especificació vàlida per al desenvolupament d'un algoritme que el resolgui. Proves formals per validar programes. Disseny basat en contractes. Precondicions, postcondicions i invariants. Anàlisi de complexitat. Paradigmes per al disseny d'algorismes. Greedy. recursivitat, backtracking, branch & bound, programació dinàmica, algoritmes probabilístics, etc. El desenvolupament d'un algoritme comença per formalitzar l'enunciat d'un problema. A partir d'aquest enunciat es dissenya un algoritme que solucioni el problema, però això no és suficient, cal considerar quant trigarà l'algorisme en donar-nos la solució. Així que ens interessa crear algoritmes el més ràpids possibles. D'aquesta manera podrem crear programes que solucionin problemes el més grans possibles en temps acceptables. Aquesta rapidesa s'aconsegueix dissenyant algoritmes que minimitzin el nombre d'operacions a realitzar per resoldre un problema i desenvolupant una implementació eficient de les operacions de l'algoritme. Això suposarà que en aquesta assignatura l'alumne adquirirà coneixements sobre algorísmica i implementació eficient d'algoritmes.

5. Prova de programes.Depuració d'errors

Totes les fases establertes en el desenvolupament del programari són importants. La manca o mala execució d'alguna pot provocar que la resta del projecte arrossegui un o diversos errors que seran determinants per a l'èxit del projecte.Una aplicació informàtica no pot arribar a les mans d'un usuari final errades, i menys si aquesta és prou visible i clara per

haver estat detectada pels desenvolupadors. Es donaria una situació de manca de professionalitat i de confiança per part dels usuaris en els desenvolupadors. Aquesta pèrdua de confiança desenvoluparia en una manca de futures oportunitats.És per això que aquesta fase del desenvolupament d'un projecte de programari es considera bàsica abans de fer la transferència del projecte a l'usuari final. Qui donaria un cotxe per construït i finalitzat si en intentar arrencar-lo no funciona?
Validació de la correctesa en el desenvolupament d'una aplicacióLa validació és l'activitat encarregada d'assegurar-se que l'aplicació informàtica obtinguda és la que s'havia determinat inicialment i és la que es volia construir. La validació confirmarà que l'aplicació funciona tal com el client ha demanat. El sistema fa el que el client vol?
La verificació és l'activitat que comprova el funcionament correcte del codi programat. Es comprovarà que tecnològicament l'aplicació informàtica s'hagi desenvolupat de manera correcta. El sistema fa el que ha de fer?

6. Documentació dels programes

6.1 LA DOCUMENTACIÓN DE PROGRAMAS

En la ejecución de un proyecto informático o un programa software se deben de seguir una serie de pasos desde que se plantea el problema hasta que se dispone del programa o del a aplicación funcionando en el ordenador. Los pasos son los siguientes:
q Análisis de factibilidad
q Análisis de requerimientos
q Diseño del sistema
q Implementación
q Validación y pruebas
q Explotación
q Mantenimiento

Cada uno de estos pasos debe de llevar asociado un documento. Estos documentos son muy importantes ya que van a regir las fases del ciclo de vida del software y se recogen los pasos seguidos en cada fase para su ejecución. No es viable la solución mostrada por algunos programadores de ir directamente a la implementación sin antes pararse en la fases 1, 2 y 3. Un trabajo deficiente en estas fases supone una mala definición del problema y por tanto el sistema no cumplirá seguramente con todos los requisitos. El diseño del sistema no será efectivo y los errores serán de difícil solución. Por lo tanto en la realización

de las prácticas será obligado cumplimentar un formulario que guiará al alumno en la fase de análisis de requisitos y de diseño.

6.2 EL DOCUMENTO DE ESPECIFICACIONES

Este documento tiene como objeto asegurar que tanto el desarrollador como el cliente tienen la misma idea sobre las funcionalidades del sistema. Es muy importante que esto quede claro ya que si no el desarrollo software no será aceptable. En el caso de este curso, es importante que el alumno y el profesor tengan la misma idea de que hay que desarrollar en la práctica, si un alumno no desarrolla lo que el profesor espera no obtendrá una nota adecuada con su expectativa. Por lo tanto es muy importante que las especificaciones del problema estén claras por ambos. Existe una normativa referente a este tipo de documento, en Ingeniería del Software I se os dará con más detalle, aquí sólo se intenta que se entienda el alcance e importancia de este documento.

Según la norma IEEE 830, un ERS debe contener los siguientes puntos: I. Introducción (Se definen los fines y los objetivos del software) A. Referencia del sistema B. Descripción general C. Restricciones del proyecto II. Descripción de la información (Descripción detallada del problema, incluyendo el HW y SW necesario) A. Representación del flujo de la información. 1. Flujo de datos 2. Flujo de control B. Representación del contenido de la información. C. Descripción de la interfaz del sistema. III. Descripción funcional (Descripción de cada función requerida, incluyendo diagramas) A. Partición funcional B. Descripción funcional 1. Narrativa de procesamiento 2. Restricciones/Limitaciones. 3. Requisitos de rendimiento. 4. Restricciones de diseño 5. Diagramas de soporte C. Descripción del control 1. Especificación del control 2. Restricciones de diseño IV. Descripción del comportamiento (comportamiento del SW ante sucesos externos y controles internos) A. Estados del sistema B. Sucesos y acciones 3 V. Criterios de validación. A. Límites de rendimiento B. Clases de pruebas C. Respuesta esperada del SW D. Consideraciones especiales VI. Bibliografía VII. Apéndice.

6.3 EL DOCUMENTO DE DISEÑO

En la fase de diseño se toman aquellas decisiones relativas a la futura implementación, se decide la estructura de datos a utilizar, la forma en que se van a implementar las distintas estructuras, el contenido de las clases (sus métodos, los atributos, ...), los objetos. También se definen las funciones, sus datos de entrada y salida, que tarea realizan, para alguna de

especial interés el algoritmo que soluciona el problema. El flujo del programa se define mediante una serie de gráficos que permiten visualizar cual es la evolución del sistema software, en caso de orientación de objetos existen el diagrama de clases, el diagrama es importante tenerlo claro para ello existen una serie de diagramas que permitan clarificar este asunto.

6.4 LA DOCUMENTACIÓN DEL CÓDIGO FUENTE

Durante la fase de implementación, cuando se está programando, es necesario comentar convenientemente cada una de las partes que tiene el programa. Estos comentarios se incluyen en el código fuente con el objeto de clarificar y explicar cada elemento del programa, se deben de comentar las clases, las variables, los módulos y en definitiva todo elemento que se considere importante. Esta documentación tiene como objeto hacer más comprensible el código fuente a otros programadores que tengan que trabajar con él, ya sea porque forman parte del grupo de desarrollo, el programa va a ser mantenido o modificado por otra persona distinta al programador inicial. También resulta muy útil durante la depuración y el mantenimiento del programa por el propio programador, al paso del tiempo las decisiones se olvidan y surgen dudas hasta en el propio programador de porqué se hicieron las cosas de una determinada manera y no de otra.

7. Entorn de desenvolupament de programes

Els entorns de desenvolupament són eines que ofereixen als programadors moltíssimes facilitats a l'hora de crear una aplicació informàtica. El mòdul "Entorns de Desenvolupament" mostra quin és el procés de desenvolupament d'una aplicació informàtica, tot indicant les característiques més importants de les eines que ajuden a aquest procediment.

1.Desenvolupament de programari

Un **programa informàtic** és un conjunt d'esdeveniments ordenats de manera que se succeeixen de forma seqüencial en el temps, un darrere l'altre.Per tant,un programa informàtic és tenir clar què és un programa. Així doncs, un programa és un conjunt d'instruccions codificades i de dades que són l'expressió completa d'un procediment, i en particular d'un algorisme, executable per un sistema informàtic.

Per convertir aquesta concepció d'una idea abstracta en un producte acabat que sigui eficaç i eficient hi haurà molts més passos, moltes tasques a fer. Aquestes tasques caldrà que estiguin ben planificades i que segueixin un guió que pot tenir en compte aspectes com:

- Analitzar les necessitats que tenen les persones que faran servir aquest programari, escoltar com el voldran, atendre a les seves indicacions...
- Dissenyar una solució que tingui en compte totes les necessitats abans analitzades: què haurà de fer el programari, quines interfícies gràfiques tindrà i com seran aquestes, quines dades s'hauran d'emmagatzemar i com es farà...
- Desenvolupar el programari que implementi tot allò analitzat i dissenyat anteriorment, fent-lo d'una forma al més modular possible per facilitar el posterior manteniment o manipulació per part d'altres programadors.
- Dur a terme les proves pertinents, tant de forma individualitzada per a cada mòdul com de forma complerta, per tal de validar que el codi desenvolupat és correcte i que fa el que ha de fer segons l'establert en els requeriments.
- Implantar el programari en l'entorn on els usuaris finals el faran servir.Aquest apartat se centrarà en el tercer punt, el desenvolupament de programari.

2. Codi font, codi objecte i codi excutable: màquines virtuals

Per crear un programa el que es farà serà crear un arxiu i escriure a un fitxer el seguit d'instruccions que es vol que l'ordinador executi. Aquestes instruccions hauran de seguir unes pautes determinades en funció del llenguatge de programació escollit. A més, haurien de seguir un ordre determinat que donarà sentit al programa escrit. Per començar n'hi haurà prou amb un editor de text simple.

El conjunt de fitxers de text resultants, on es troben les instruccions, es diu que contenen el **codi font.**
El codi objecte és el codi font traduït (pel compilador) a codi màquina, però aquest codi encara no pot ser executat per l'ordinador.
El codi executable és la traducció completa a codi màquina, duta a terme per l'enllaçador (en anglès, linker). El codi executable és interpretat directament per l'ordinador.

Exemples d'entorn integrat de desenvolupament

Com a exemples d'IDE multiplataforma es poden trobar, entre molts d'altres:

- Eclipse, projecte multiplataforma (Windows, Linux, Mac) de codi obert, fundat per IBM el novembre de 2001, desenvolupat en Java.
- Netbeans, projecte multiplataforma (Windows, Linux, Mac, Solaris) de codi obert, fundat per Sun Microsystems el juny de 2000, desenvolupat en Java.
- Anjuta DevStudio, per al GNU/Linux, creat per Naba Kumar el 1999.JBuilder, eina multiplataforma (Windows, Linux, Mac), propietat de l'empresa Borland, apareguda el 1995. La versió 2008 incorpora tres edicions (Enterprise –de pagament–, Professional –de pagament– i Turbo –gratuïta-).
- JDeveloper, eina multiplataforma (Windows, Linux, Mac) gratuïta, propietat de l'empresa Oracle, apareguda el 1998, inicialment basada en JBuilder però desenvolupada des de 2001 en Java.

A part dels que hem esmentat, existeixen molts altres IDE, molt coneguts, per a determinades plataformes, com per exemple:

- Visual Studio
- Dev-Pascal
- Dev-C++
- MonoDevelop

Avui dia els entorns de desenvolupament proporcionen un marc de treball per a la majoria dels llenguatges de programació existents en el mercat.

Instal·lació d'un entorn de desenvolupament. Eclipse

Cada programari i cada entorn de desenvolupament té unes característiques i unes funcionalitats específiques.

Cal que tingueu presents els següents conceptes:

- JVM (Java Virtual Machine, màquina virtual de Java) s'encarrega d'interpretar el codi de bytes i generar el codi màquina de l'ordinador (o dispositiu) en el qual s'executa l'aplicació. Això significa que ens cal una JVM diferent per a cada entorn.
- JRE (Java Runtime Environment) és un conjunt d'utilitats de Java que inclou la JVM, llibreries i el conjunt de programari necessari per executar les aplicacions client de Java, així com el connector per tal que els navegadors d'internet puguin executar les applets.
- JDK (Java Development Kit, kit de desenvolupament de Java) és el conjunt d'eines per a desenvolupadors; conté, entre altres coses, el JRE i el conjunt d'eines necessàries per compilar el codi, empaquetar-lo, generar documentació...

El procés d'instal·lació consisteix en els següents passos:
Descarregar, instal·lar i configurar el JDK.
Descarregar i instal·lar un servidor web o d'aplicacions.
Descarregar, instal·lar i configurar Eclipse.
Configurar JDK amb l'IDE d'Eclipse.
Configurar el servidor Apache Tomcat en Eclipse.
En cas de ser necessari, instal·lació de connectors.
En cas de ser necessari, instal·lació de nou programari, com per exemple WindowBuilder.