Módulo Profesional 05:
Entornos de desarrollo

UF1 Actividad A

CICLO FORMATIVO DE GRADO SUPERIOR EN
DESARROLLO DE APLICACIONES MULTIPLATAFORMA
MODALIDAD ONLINE

ALUMNA:

MARÍA LAURA RONDINA IOBBI

Descripción de la actividad

En la siguiente actividad teórica se valorarán los conceptos más importantes de la actividad.

Desarrollo de la actividad

Dar respuesta a los siguientes ejercicios teóricos en el mismo documento y entregar por el campus. Poner nombre al documento y entregar en formato Word/pdf.

1. ¿Cuáles son los diferentes estados por los que pasa el código desde que el programador define el problema a resolver hasta que se entrega el producto que se soluciona? [1.0 puntos]

Desde que el programador plantea su solución al problema a resolver, hasta que entrega el producto final, ocurren varias fases o "estados" del código:

1) **Código Fuente**: Es el conjunto de líneas de texto o instrucciones relativas a los pasos que debe seguir la computadora para ejecutar un programa, escritas en un determinado lenguaje de programación, con una sintaxis "cercana al lenguaje natural", comprensible para aquellas personas con conocimientos de la tecnología que se esté utilizando.

2) **Código Objeto:** En el primer estado anterior el código fuente no es directamente ejecutable por la computadora, sino que debe ser traducido a otro lenguaje o código binario; así será más fácil para la máquina interpretarlo. Esto es lo que se conoce como lenguaje máquina o código objeto, que es el que resulta de la compilación del código fuente, y que como resultado de este proceso de traducción puede ser ejecutado por el hardware de la computadora. Para traducir el código fuente a código objeto, se usan los llamados compiladores, ensambladores, intérpretes y otros sistemas de traducción.

3) **Código ejecutable:** Para llegar a la tercera fase, el enlazador dará sentido a los objetos creados, convirtiendo al código objeto en código ejecutable, que es aquel que corresponde a unidades de programas donde la computadora puede realizar las instrucciones compiladas que tendrán enlazadas una o varias bibliotecas. En definitiva, se trata de Uno o varios códigos objetos enlazados que pueden ser ejecutados por un ordenador, tras ser interpretados o compilados. El código ejecutable se encuentra empaquetado y listo para ser utilizado en cualquier computadora. El beneficio de tener el código ejecutable, es que se puede saber que la compilación fue realizada correctamente y que el programa, si no tiene errores de manejo, puede funcionar correctamente, porque está libre de errores de variables, signos y demás.

Por último, las **máquinas virtuales** permitirán que un mismo programa pueda ser ejecutado, sin necesidad de cambios, en diferentes sistemas operativos.

2. ¿Cuántos tipos de lenguaje de programación existen? Dar una breve definición de cada uno de ellos. [1.0 puntos]

Los lenguajes de programación se pueden clasificar en tres grandes grupos en función de su forma de ejecución y su aparición histórica:

1) **Lenguajes de Procedimiento (Procedurales)**:
Un programa de procedimiento contiene una lista de acciones a realizar siempre en el mismo orden para conseguir el objetivo para el que el programa fue creado. Se caracterizan por un punto de comienzo, un proceso y un final. Los procedimientos pueden ser funciones, rutinas o métodos. Son ejemplos Basic y Fortran.

2) **Lenguajes de Programación Estructurada**:
La programación estructurada es un paradigma de programación orientado a mejorar la claridad, calidad y tiempo de desarrollo de un programa de computadora recurriendo únicamente a subrutinas y tres estructuras básicas: secuencia, selección (*if* y *switch*) e iteración (bucles *for* y *while*); asimismo, se considera innecesario y contraproducente el uso de la instrucción de transferencia incondicional (GOTO), que podría conducir a *código espagueti*, mucho más difícil de seguir y de mantener, y fuente de numerosos errores de programación. Con este tipo de lenguajes

se pretende romper la estructura monolítica de los programas en módulos más pequeños y manejables, así como diferenciar entre variables globales y locales desde un módulo concreto. Son ejemplos: C y Pascal.

3) **Lenguajes de Programación Orientada a Objetos (POO)**:
La POO es un paradigma surgido en los años 1970, que utiliza objetos como elementos fundamentales en la construcción de la solución. Se trata de una nueva forma de pensar acerca del proceso de descomposición del problema y de desarrollo de soluciones, que propone una serie de recursos basados en entidades del mundo real para la construcción de estructuras y soluciones. Es una forma especial de programar más cercana a cómo expresaríamos las cosas en la vida real que otros tipos de programación. Son ejemplos: Java, C++, C# y Visual Basic.

3. Menciona dos lenguajes de programación orientados a objetos y menciona la diferencia que tienen con respecto a los lenguajes de procedimiento y estructurados. [1.0 puntos]

Dos de los lenguajes de programación orientados a objetos más comunes son Java y C++. Las diferencias con respecto a los lenguajes de procedimiento y estructurados se mencionan a continuación.

· La POO es **más moderna**, es una evolución de la programación estructurada que plasma en el diseño de una familia de lenguajes conceptos que existían previamente con algunos nuevos.

· En POO, el análisis está o**rientado a objetos**, mientras que en el enfoque estructurado, está orientado a los procesos del sistema. La programación procedimental no necesita objetos; tiene procedimientos que podrían ser estructuras de datos, rutinas y subrutinas.

· La **nomenclatura** difiere. En programación de procedimental, las funciones se denominan*"procedimientos"*, mientras que en POO son *"métodos"*. En cuanto a las estructuras de datos, la programación procedimental los etiqueta como *"registros"* mientras que POO usa*"objetos"*.La Programación procedimental usa una *llamada de procedimiento* para llamar a una función, mientras que POO utiliza una *llamada de mensaje* para solicitar acciones de objetos.

· La POO se basa en una **nueva forma de pensar los problemas**, una forma especial de programar, más cercana a como expresaríamos las cosas en la vida real que otros tipos de programación, y que se centra en el uso de clases y objetos. La programación estructurada consta de una estructura donde se va ejecutando paso a paso y este debe de tener una secuencia y una lógica para que su función sea eficiente, y que pretende resolver un problema de principio a fin en una sola estructura de código.

· La POO se basa en lenguajes que soportan sintáctica y semánticamente la unión entre los tipos abstractos de datos y sus operaciones (a esta unión se la suele llamar **clase**).

· La POO incorpora en su entorno de ejecución mecanismos tales como el polimorfismo y el envío de mensajes entre objetos. En POO, se puede lograr un ***polimorfismo de subtipo***, mediante el cual una función escrita para los elementos de un tipo de datos (*supertipo*) se puede hacer funcionar en los elementos de otro tipo de datos relacionado (*subtipo*). La programación estructurada no tiene esta habilidad.

· La POO resuelve el problema identificando los actores que tienen participación en el problema e identificando también sus acciones, y con esta información se crean los **objetos**. Al tener creados los objetos sólo es ponerlos a interactuar entre ellos.

· La característica más distinguible de la POO es la ***herencia***, a través de la cual el código se puede reutilizar y extender sin cambiar el código existente. Los objetos nuevos son capaces de "heredar" las propiedades de los objetos más antiguos. La programación procedimental no admite la herencia, porque sólo se puede aplicar a los objetos.

· A diferencia de su contraparte, la POO utiliza la **Encapsulación**: forma una cápsula imaginaria que envuelve los datos y métodos, protegiéndolos de la interferencia externa. El código se puede escribir para restringir el uso de datos fuera de la cápsula en la que se emplea.

· La POO facilita la **Reusabilidad**: Cuando hemos diseñado adecuadamente las clases, se pueden usar en distintas partes del programa y en numerosos proyectos.

· La POO facilita la **Mantenibilidad**: Debido a la sencillez para abstraer el problema, los programas orientados a objetos son más sencillos de leer y comprender, pues nos permiten ocultar detalles de implementación dejando visibles sólo aquellos detalles más relevantes.

· La POO facilita la **Modificabilidad:** La facilidad de añadir, suprimir o modificar nuevos objetos nos permite hacer modificaciones de una forma muy sencilla.

· La POO aumenta la **Fiabilidad**. Al dividir el problema en partes más pequeñas podemos probarlas de manera independiente y aislar mucho más fácilmente los posibles errores que puedan surgir.

DIFICULTADES DE LA PROGRAMACIÓN ESTRUCTURADA/PROCEDIMENTAL:

· Nuestra imagen del mundo se apoya en los seres, a los que asignamos nombres sustantivos, mientras la programación clásica se basa en el comportamiento, representado usualmente por verbos.

· Es difícil modificar y extender los programas, pues suele haber datos compartidos por varios subprogramas, que introducen interacciones ocultas entre ellos.

· Es difícil mantener los programas. Casi todos los sistemas informáticos grandes tienen errores ocultos, que no surgen a la luz hasta después de muchas horas de funcionamiento.

· Es difícil reutilizar los programas. Es prácticamente imposible aprovechar en una aplicación nueva las subrutinas que se diseñaron para otra.

· Es compleja la coordinación y organización entre programadores para la creación de aplicaciones de media y gran envergadura.

· En la programación orientada a objetos pura no deben utilizarse llamadas de subrutinas, únicamente mensajes. Por ello, a veces recibe el nombre de programación sin CALL, igual que la programación estructurada se llama también programación sin GOTO. Sin embargo, no todos los lenguajes orientados a objetos prohíben la instrucción CALL (o su equivalente), permitiendo realizar programación híbrida, imperativa y orientada a objetos a la vez.

4. Menciona 3 Características principales de un lenguaje de programación orientado a objetos. [1.0 puntos]

En este tipo de lenguajes, el programador diseña la estructura de datos y el tipo de operaciones que acepta dicha estructura. Al conjunto de estructura de dato y operación se le llama Objeto. El programa ya no es una lista de acciones de orden fijo sino un conjunto de objetos (datos y posibles operaciones que puede realizar) que cooperan entre ellos para conseguir el objetivo.

Con la POO, se introducen nuevos conceptos que se aplican a los objetos mencionados y que destacan entre sus características principales:

- **Clase:** Modelo desde el que los objetos son creados(conjunto de variables y métodos).El objeto creado puede posteriormente ser modificado mediante nuevas estructuras de datos y métodos.

- **Encapsulación:** Los datos y métodos de una estructura se protegen de aquellos accesos que no sigan las normas especificadas para el objeto. Asimismo el programador decide qué información del objeto puede ser compartida por otros objetos.

- **Polimorfismo:** Objetos de diferente tipo capaces de recibir el mismo mensaje y responder de diferente modo.

A estas características pueden añadirse, además, otras propiedades elementales de la POO que son:

-Herencia: Las clases se relacionan entre sí y forman una jerarquía de clasificación. Los objetos heredan las propiedades y el comportamiento de todas las clases a las que pertenecen y pueden compartir (y extender) su comportamiento sin tener que volver a implementarlo.
-Modularidad: Propiedad que permite subdividir una aplicación en partes más pequeñas (módulos), cada una de las cuales debe ser tan independiente como sea posible de la aplicación en sí y de las restantes partes.
-Ocultación: Cada objeto está aislado del exterior, es un módulo natural, y cada tipo de objeto expone una "interfaz" a otros objetos que especifica cómo pueden interactuar con los objetos de la clase. El aislamiento protege a las propiedades de un objeto contra su modificación por quien no tenga derecho a acceder a ellas; solamente los propios métodos internos del objeto pueden acceder a su estado.

5. Menciona las etapas de transformación por las que pasa el código objeto hasta llegar a ser código ejecutable. [1.0 puntos]

Los elementos y fases implicados en la obtención de un código ejecutable a partir del código fuente inicial son los siguientes:

1) **Fichero Fuente**: La creación de uno o más ficheros fuente en los que son almacenadas las instrucciones en un lenguaje de alto nivel y que puede realizarse con un sencillo procesador de textos.

2) **Preprocesamiento:** Se realiza un rastreo del código fuente para detectar todos los posibles errores de entrada de código, como pueden ser: paréntesis sin cerrar, asignaciones de variables inválidas, etcétera.

3) C**ompilador:** Este programa lee el fichero fuente y lo traduce en un tipo de código comprensible para el ordenador en el que se ha realizado la compilación. Cada tipo de ordenador tiene su propio compilador. Esta traducción a un código entendible para la computadora recibe el nombre de Código Objeto, y no es aún ejecutable.

4) **Enlazador:** Después de que el compilador haya creado todos los objetos necesarios, el enlazador los procesará junto con las librerías de rutinas, funciones y datos, que el propio lenguaje aporta al proceso de desarrollo para la creación del ejecutable.

6. ¿Cómo se llama el programa traductor de código fuente a código objeto? [1.0 puntos]

El programa traductor de código fuente a código objeto recibe el nombre de **Compilador**. En informática, un compilador es un tipo de traductor que transforma un programa entero de un lenguaje de programación (llamado código fuente) a otro. Usualmente el lenguaje objetivo es código máquina, aunque también puede ser traducido a un código intermedio (*bytecode*) o a texto. A diferencia de los intérpretes, los compiladores reúnen diversos elementos o fragmentos en una misma unidad (un programa ejecutable o una librería), que puede ser almacenada y reutilizada. Este proceso de traducción se conoce como *compilación*.

La construcción de un compilador involucra la división del proceso en una serie de fases que variarán con su complejidad. Generalmente estas fases se agrupan en dos tareas: el análisis del programa fuente y la síntesis del programa objeto.

- **Análisis:** se trata de la comprobación de la corrección del programa fuente. Incluye las fases correspondientes al *análisis léxico* (que consiste en la descomposición del programa fuente en componentes léxicos), *análisis sintáctico* (agrupación de los componentes léxicos en frases gramaticales) y *análisis semántico* (comprobación de la validez semántica de las sentencias aceptadas en la fase de análisis sintáctico).

- **Síntesis:** su objetivo es la generación de la salida expresada en el lenguaje objeto y suele estar formado por una o varias combinaciones de fases de generación de código (normalmente se trata de código intermedio o de código objeto) y de optimización de código (en las que se busca obtener un programa objetivo lo más eficiente posible en lo relativo a tiempo de ejecución, espacio durante ejecución, espacio para ser almacenado fuera de ejecución, etc).

7. La fase en que los objetos del proyecto pasan a explotación, ¿se considera dentro de las fases del desarrollo de una aplicación? ¿Si, no y por qué? [1.0 puntos]

El desarrollo de una aplicación informática debe pasar necesariamente por una serie de fases que ayudarán a realizar la entrega del producto en el plazo previsto, con la calidad pactada y con el coste razonablemente calculado. Estas fases se resumen a continuación:

▪ **Planificación del proyecto. Estudio de viabilidad.**
- Documento del presupuesto. Después de la planificación, el responsable del proyecto debe valorarlo económicamente.

- Firma del documento de iniciación del proyecto.
- Análisis del sistema y recogida de requerimientos
- Diseño del sistema.
- **Codificación**
- **Pruebas técnicas**
- **Pruebas de usuario.**
- **Documentación.**
- **Implementación en entorno de explotación (Real).**
- **Plan de mantenimiento**

Esto implica que la fase en la que los objetos del proyecto pasan a explotación (es decir, la antes mencionada Fase de Implementación en entorno de explotación (Real)), SÍ se considera dentro de las fases del desarrollo de una aplicación. Aunque previamente se han llevado a cabo Pruebas técnicas (en las que el programador ha realizado las pruebas necesarias para demostrar que sus objetos realizan correctamente las tareas demandadas) y Pruebas de Usuario (en las que en usuario debe realizar las pruebas que crea convenientes en un entorno con datos ficticios y protegido de la explotación en real, para dar el visto bueno al desarrollo), la implementación en un entorno de explotación real abrirá un necesario periodo de ventana de monitorización de la instalación y de posibles ajustes o incluso vuelta atrás por mal funcionamiento.

8. ¿Define qué es una máquina virtual y para que se puede utilizar? [1.0 puntos]

Una **máquina virtual** es un programa de ordenador que emula el funcionamiento de otro sistema informático completo, pudiendo realizar en él todas las operaciones que haríamos en un hardware real. Las máquinas virtuales permiten la instalación de diferentes sistemas operativos mediante la utilización de la misma máquina física. Existen varios tipos:

1) DE SISTEMA. Emulan un ordenador real:

-**De Replicación**: Son una réplica normalmente simplificada de la máquina real sobre la que funciona. Ejemplo:VWWare,Hyper-V,VirtualBox.

-**De Emulación**: Recrean el funcionamiento de máquinas cuyo hardware es diferente de aquel que las soporta. Ejemplo:Emu48 .

2) DE PROCESO. No emulan un hardware real. Crean un entorno abstracto con el objetivo de ejecutar sobre él aplicaciones que no dependan de la máquina física sobre la que trabajan. Ejemplo: Java Virtual Machine.

En la actualidad, se ha generalizado la virtualización de los equipos de hardware porque reduce coste y riesgos, al tiempo que aumenta la calidad y el aprovechamiento de los equipos; permite crear diferentes sistemas operativos totalmente interdependientes entre ellos en una misma máquina y permite un mejor aislamiento y administración de los recursos. Entre sus desventajas, destacan el hecho de que un fallo en el software supondría la caída de todos los sistemas virtuales ejecutados en esa máquina, y que para dar un servicio óptimo a los equipos virtualizados es necesario un hardware potente y por lo tanto, más caro.

9. Menciona tres de las principales funciones habilitadas en un entorno de desarrollo. [1.0 puntos]

El menú desplegable del entorno de desarrollo nos ofrece las principales actividades que es posible realizar. Tres de las funciones principales habilitadas son:

-Entrada de Codificación (normalmente en un lenguaje de alto nivel) mediante un editor de texto.

-Ejecución del programa en pruebas, línea a línea o por secciones, lo que facilita la rápida detección y depuración de errores.

-Asistencia en la documentación y mantenimiento posteriores.

10. Menciona y define brevemente las áreas en las que se divide una pantalla básica de desarrollo. [1.0 puntos]

En la pantalla principal del entorno de desarrollo se pueden identificar varias áreas de trabajo

Área 1: Menú principal. Es la sección desde donde se realizan las principales acciones del entorno(ficheros, depuración y contrl del proyecto). En el ejemplo mostrado de IDE de NetBeans, localizamos opciones como File,Edit,View, Navigate, etc.

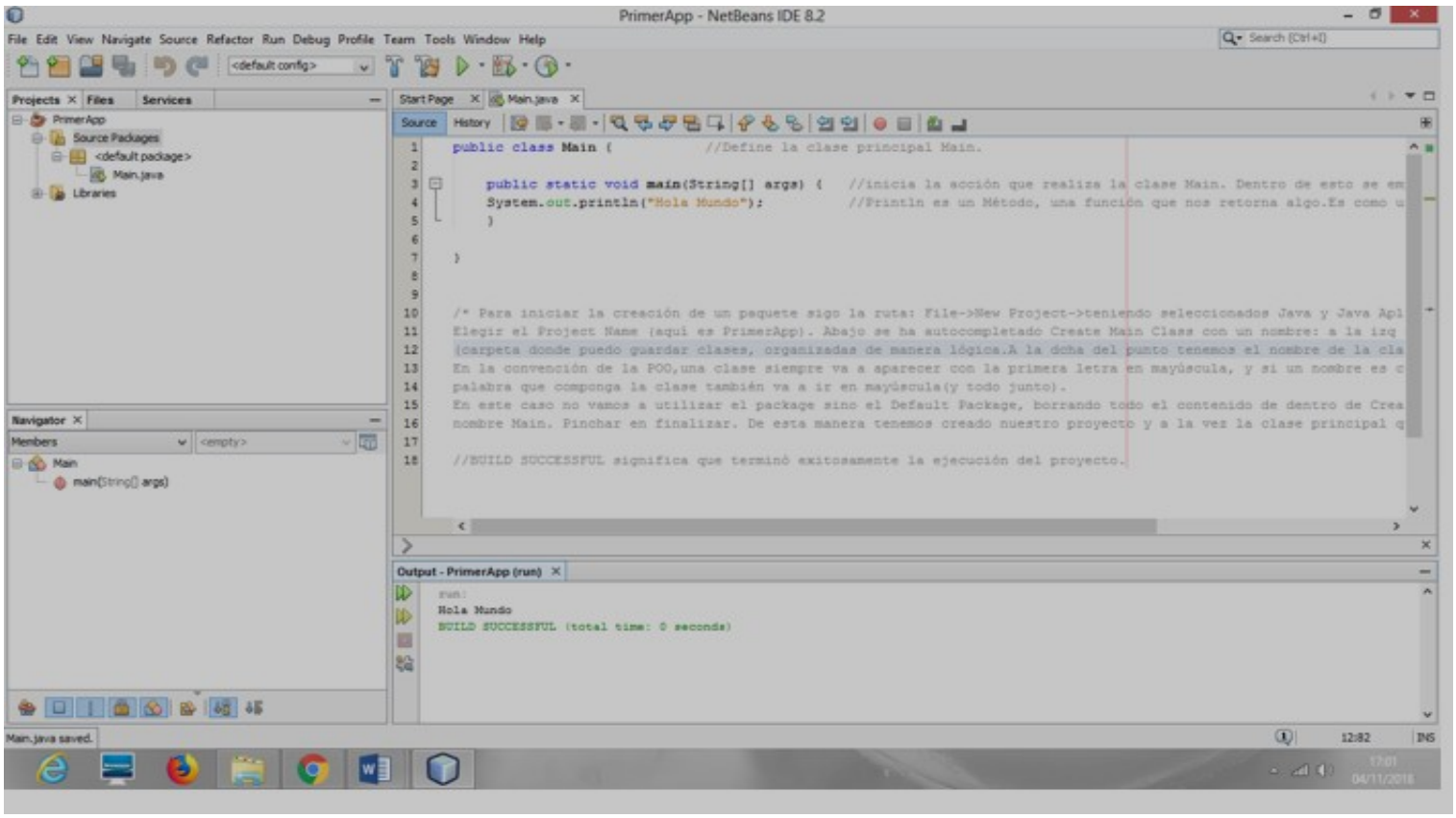
Área 2:Ventana de Proyecto. Muestra una visión de conjunto del proyecto (ficheros, librerías necesarias para el código entrado, etc). En el ejemplo, es el panel superior izquierdo visualizado en la pestaña Projects.

Área 3: Ventana de Navegación. Permite acceder de forma rápida a los elementos de las clases que tengamos seleccionadas. En el ejemplo, es el panel inferior izquierdo de la pestaña Navigator.

Área 4: Ventana de Edición. Editor de texto a través del cual entramos en el código del desarrollo de nuestro proyecto. En el ejemplo, es el panel central donde se ha escrito un código para ejecutar la salida de texto “Hola Mundo”.

Área 5: Ventada de Salida/Tareas. Muestra los errores de compilación e información seleccionada por el programador mediante palabras clave del código. En el ejemplo se localiza al final del IDE, con el nombre Output.

En la siguiente página se puede apreciar una imagen de un entorno de desarrollo comúnmente utilizado, NetBeans IDE:


PrimerApp - NetBeans IDE 8.2
File Edit View Navigate Source Refactor Run Debug Profile Team Tools Window Help
Search (Ctrl+I)
<default config>
Projects
Files
Services
PrimerApp
Source Packages
<default package>
Main.java
Libraries
Navigator
Members
<empty>
Main
main(String[] args)
Start Page
Main.java
Source
History
1 public class Main { //Define la clase principal Main.
2
3 public static void main(String[] args) { //inicia la acción que realiza la clase Main. Dentro de esto se en
4 System.out.println("Hola Mundo"); //Println es un Método, una función que nos retorna algo.Es como u
5 }
6
7 }
8
9
10 /* Para iniciar la creación de un paquete sigo la ruta: File->New Project->teniendo seleccionados Java y Java Apl
11 Elegir el Project Name (aquí es PrimerApp). Abajo se ha autocompletado Create Main Class con un nombre: a la izq
12 (carpeta donde puedo guardar clases, organizadas de manera lógica.A la dcha del punto tenemos el nombre de la cla
13 En la convención de la POO,una clase siempre va a aparecer con la primera letra en mayúscula, y si un nombre es c
14 palabra que componga la clase también va a ir en mayúscula(y todo junto).
15 En este caso no vamos a utilizar el package sino el Default Package, borrando todo el contenido de dentro de Crea
16 nombre Main. Pinchar en finalizar. De esta manera tenemos creado nuestro proyecto y a la vez la clase principal q
17
18 //BUILD SUCCESSFUL significa que terminó exitosamente la ejecución del proyecto.
Output - PrimerApp (run)
run:
Hola Mundo
BUILD SUCCESSFUL (total time: 0 seconds)
Main.java saved.
12:82
INS