PARADIGMAS DE PROGRAMACIÓN ORIENTADA A OBJETOS I

UNIVERSIDAD DE LA SIERRA SUR

Universidad de la Sierra Sur

Licenciatura en Informática

Planeación académica

Identificación de la asignatura				
Asignatura:	Paradigmas de Programación I		Clave:	6041
Semestre:	Cuarto (21-22B)		Grupo:	406
Créditos:	10	Antecedente curricular:		N/A
Profesor-Investigador:	M.T.C.A. Rolando Pedro Gabriel			
Correo electrónico:	rolando.pedro.gabriel@gmail.com			
Asesorías:	Lunes – Viernes	11:00 - 12:00		Plataforma Institucional

Evaluación de la asignatura		
Exámenes Parciales (3)	50 %	Los exámenes serán los días: 1. Del 29 de marzo al 05 de Abril de 2022 2. Del 09 al 16 de Mayo de 2022 3. Del 10 al 17 de Junio de 2022
Examen Ordinario	50 %	Del 23 al 30 de Junio de 2022.
Total evaluación	100 %	

Parámetros para las evaluaciones parciales		
Examen escrito	70 %	El examen teórico-práctico deberá ser presentado en forma escrita.
Ejercicios, tareas y trabajos de investigación sobre los temas del parcial correspondiente.	30 %	Para cada ejercicio, tarea y trabajo se especificarán los requisitos y criterios de evaluación correspondientes en el momento de asignarlos. Las lecturas recomendadas serán utilizadas para su análisis y posterior discusión en el aula. Serán cancelados trabajos: ▪ Copiados o duplicados de Internet o del trabajo de otro compañero; ▪ Contenidos del documento que no tengan relación con lo solicitado. Los trabajos se verán afectados en su calificación cuando: ▪ Entregados fuera de tiempo; ▪ No cuenten con los elementos solicitados: portada, índice de contenido (en caso de requerirlo), formateo adecuado del documento, numeración adecuada de páginas, conclusión (en caso de requerirlo); ▪ Presente faltas de ortografía.
Total evaluación	100 %	

Parámetros para la evaluación ordinaria		
Examen escrito	50 %	El examen teórico-práctico deberá ser presentado en forma escrita.
Proyecto de programación en el lenguaje de programación Java.	50 %	El alumno trabajará en el desarrollo de un proyecto empleando los conocimientos aprendidos durante el curso.
Total evaluación	100 %	

Normas y políticas institucionales	
Puntualidad:	Asistir puntualmente a clases en el horario asignado.
Retardos:	Al tercer retardo (del minuto 1 al 5), este último se considerará como una falta.

Faltas:	Si el alumno tiene más del 15% de inasistencias, no tendrá derecho al examen parcial correspondiente.
Derechos:	Revisar artículos del 25 al 28 del Reglamento de Alumnos de la Universidad de la Sierra Sur. Cualquier situación no prevista comunicarse con el profesor y la Jefatura de Carrera correspondiente.
Obligaciones:	Revisar artículo 29 del Reglamento de Alumnos de la Universidad de la Sierra Sur.
Disciplina:	Revisar artículos del 30 al 42 del Reglamento de Alumnos de la Universidad de la Sierra Sur.
Justificantes:	Expedidos por el Departamento de Servicios Escolares, presentarlos a la brevedad (Artículo 51 del Reglamento de Alumnos).
Redondeo de notas:	Se califica considerando un decimal (sin redondeo): si se tiene 7.56, entonces será considerado como 7.5 (Artículo 52 del Reglamento de Alumnos de Licenciatura).
Ética escolar:	En caso de comprobarse plagio, falsificación o copia de trabajo, examen o actividad en cuestión, el alumno será sancionado con la cancelación del mismo. En caso de reiterar, será anulada la evaluación del periodo (parcial u ordinario) correspondiente y será enviado reporte al Departamento de Servicios Escolares.
Medidas sanitarias por contingencia:	Uso correcto de cubrebocas (cubriendo desde nariz hasta mentón). Pasar por el filtro sanitario para ingresar a la Universidad. Mantener una distancia mínima de 1.5 m entre personas. Uso de careta que permita la adecuada ventilación. Desinfección o lavado de manos con agua y jabón de manera frecuente. Evitar tocarse ojos, nariz y boca. Evitar reuniones fuera de la Universidad. Evitar acudir a los cubículos de los profesores. En caso de presentar síntomas como: escurrimiento nasal, dolor de cabeza, cuerpo cortado, tos, fiebre, alteración o dificultad para reconocer olores o sabores, o diarrea, no presentarse en las instalaciones de la Universidad y notificar de manera inmediata a Jefatura de Carrera.

Bibliografía básica:

No.	Autor(es)	Título del libro	Año de Publicación	Edición	Editorial	País
1	Déléchamp, F., Laugié, H.	Java y Eclipse Desarrolle una aplicación con Java y Eclipse.	2016	*	ENI	España
2	Burnette, Ed.	Eclipse IDE Pocket Guide: Using the Full-Featured IDE.	2005	1ª	O´REILLY	USA
3	Deitel, Harvey M.	Como programar en Java	2012	9ª	Pearson educación	México
4	Joyanes Aguilar, Luis.	Programación en C,C++, Java Y UML	2010	*	McGraw Hill	México
5	Eckel, Bruce.	Piensa en Java	2007	*	Pearson	España
6	Ceballos Sierra, Francisco Javier.	JAVA 2: Curso de programación	2003	3ª	Alfaomega Ra-Ma	España
7	Joyanes Aguilar, L., Zahonero Martínez, I.	Programación en java 6: algoritmos y programación orientada a objetos e interfaz gráfica de usuario.	2011	*	McGraw Hill	España
8	Joyanes Aguilar, Luis	JAVA 2: Manual de programación	2011	1ª	McGraw Hill	España

Bibliografía de consulta:

9	Froufe Quintas, Agustín	Java 2: Manual de usuario y tutorial	2006	4ª	Alfa omega	México
10	Clay Richardson, W.	Profesional Java 2 v5.0	2005	*	Anaya Multimedia	España

11	Joyanes Aguilar, Luis	Programación en java 2: algoritmos, estructuras de datos.	2002	*	McGraw Hill	España
12	Rodríguez Sala, J. J.	Introducción a la programación. Teoría y práctica	2003	*	Club Universitario	España

Software a utilizar:	
Java	Lenguaje de programación Orientado a Objetos.
eclipse	Entorno de Desarrollo Integrado para la programación con el lenguaje Java.
UML	Lenguaje de Modelado Unificado.
Git	Es un software de control de versiones.
GitHub	Es un portal creado para alojar el código de las aplicaciones de cualquier desarrollador.

Contenido temático de la asignatura	
Objetivo general del curso:	Proporcionar los fundamentos de los paradigmas de programación; haciendo énfasis en el paradigma orientado a objetos para que sea capaz de diseñar e implementar programas en algún lenguaje de programación.

Temas	Objetivos
UNIDAD 1. CONCEPTOS GENERALES SOBRE PARADIGMAS DE PROGRAMACIÓN. (5 horas efectivas en aula y 5 horas de estudio independiente)	El alumno conocerá y diferenciará distintos paradigmas de Programación.
1.1. Introducción 1.1.1. Definición de paradigma 1.1.2. Lenguajes de programación	
1.2. Paradigmas de programación 1.2.1. Imperativa 1.2.2. Funcional 1.2.3. Lógica 1.2.4. Orientada a objetos 1.2.5. Orientada a eventos 1.2.6. Otros paradigmas	
UNIDAD 2. PARADIGMA ORIENTADO A OBJETOS. (12 horas efectivas en aula y 12 horas de estudio independiente)	El alumno aprenderá los fundamentos del Paradigma Orientado a Objetos.
2.1. Beneficios del paradigma orientado a objetos sobre otros paradigmas	
2.2. Elementos primordiales 2.2.1. Clases y objetos 2.2.2. Abstracción 2.2.3. Encapsulamiento 2.2.4. Modularidad 2.2.5. Jerarquía y herencia 2.2.6. Polimorfismo	
2.3. Ingeniería de software orientada a objetos	
2.4. Diseño de clases empleando UML	
UNIDAD 3. INTRODUCCIÓN A LA TECNOLOGÍA DE UN LENGUAJE ORIENTADO A OBJETOS (4 horas efectivas en aula y 4 horas de estudio independiente)	El alumno aprenderá las características más importantes para trabajar con el Lenguaje de Programación Java.
3.1. Conceptos generales sobre el lenguaje	
3.2. Características del lenguaje	
3.3. Compilación y ejecución de programas	
UNIDAD 4.DATOS, OPERADORES Y SENTENCIAS DE CONTROL (10 horas efectivas en aula y 10 horas de estudio independiente)	El alumno utilizará los distintos tipos de datos empleándolos en la construcción de expresiones y aprenderá las formas de
4.1. Datos 4.1.1. Tipos de datos	

4.1.2. Identificadores 4.1.3. Arreglo unidimensional (vectores) 4.1.4. Arreglo bidimensional 4.1.5. Cadenas de caracteres 4.2. Operadores y expresiones 4.2.1. Prioridad de operadores 4.2.2. Evaluación de expresiones 4.2.3. Promoción de tipos de dato 4.2.4. Conversiones de tipos de datos 4.3. Estructuras condicionales y cíclicas 4.4. Definición de clases 4.4.1. Atributos 4.4.2. Métodos 4.4.3. Instanciación 4.5. Modificadores de acceso 4.6. Composición 4.7. Ámbito de los atributos y métodos. 4.8. Auto-referencia (this)	accesarlos.
UNIDAD 5. MÉTODOS Y MENSAJES. (10 horas efectivas en aula y 10 horas de estudio independiente) 5.1. El método como elemento de la comunicación 5.1.1. Concepto de parámetro 5.1.2. Parámetros de salida y de entrada 5.2. Declaración de métodos 5.3. Llamadas a métodos (mensajes) 5.4. Tipos de métodos 5.4.1. Métodos constantes y estáticos 5.4.2. Métodos normales y volátiles 5.5. Forma de pasar argumentos 5.6. Devolver un valor desde un método 5.7. Sobrecarga de métodos	El alumno comprenderá la importancia de los métodos y mensajes y los pondrá en práctica.
UNIDAD 6.CONSTRUCTORES (5 horas efectivas en aula y 5 horas de estudio independiente) 6.1. Conceptos de constructor. 6.2. Declaración de métodos constructores 6.2.1. Constructor sin parámetros 6.2.2. Constructor con parámetros 6.2.3. Sobrecarga de constructores 6.2.4. Constructor copia	El alumno utilizará y creará constructores en sus programas desarrollados.
UNIDAD 7. HERENCIA (12 horas efectivas en aula y 12 horas de estudio independiente) 7.1. Introducción a la herencia 7.2. Herencia simple 7.3. Herencia múltiple 7.4. Clase base y clase derivada 7.4.1. Definición 7.4.2. Declaración 7.4.3. Uso de métodos de la clase base 7.5. Importancia de la encapsulación 7.6. Sobreescritura de métodos 7.7. Constructores y herencia 7.8. Introducción al polimorfismo	El alumno entenderá la importancia de la Herencia y aprenderá a implementarla en sus programas en Java.
UNIDAD 8.INTERFACES GRÁFICAS DE USUARIO (GUI) (15 horas efectivas en aula y 15 horas de estudio independiente) 8.1. Contenedores. 8.2. Componentes 8.3. Administradores de diseño de GUI 8.4. Elementos gráficos 8.5. Cuadros de diálogo 8.6. Control de eventos	El alumno creará programas que cuenten con interfaces gráficas para comunicarse con el usuario.

M.T.C.A. Rolando Pedro Gabriel

Elaboró

M.T.E. Everardo de Jesús Pacheco Antonio

Vo. Bo.
Jefe de Carrera

LICENCIATURA EN INFORMÁTICA

Misión LI:
Formar profesionales capaces de detectar, proponer y desarrollar soluciones informáticas eficientes, brindando a los estudiantes una educación superior de alta calidad, fomentando el desarrollo científico a través de la investigación y promoviendo el desarrollo de la región.

Visión LI:
Lograr a través de sus egresados un amplio reconocimiento en el área de Informática, mostrando ser un programa generador de profesionales comprometidos con su entorno y capaces de mejorar la situación económica, social y tecnológica de la región sin descuidar el ambiente y la cultura, fortaleciendo la integración de los egresados en el ámbito global.

1. Se formarán equipos para desarrollar una aplicación de escritorio, en el que los alumnos integren gradualmente los conocimientos que van adquiriendo durante el curso.

2. Cada equipo determinará la temática a desarrollar.

3. Cada equipo deberá integrar su aplicación en un repositorio en la nube, por ejemplo; github, gitlab, etc.

4. El producto a desarrollar es una aplicación de escritorio.

5. Durante el semestre se revisarán elementos básicos de Java y elementos gráficos para la interfaz de usuario. Ambas herramientas se utilizarán para desarrollar la aplicación de escritorio mencionado en el punto 4.

6. Dependiendo de la cantidad de integrantes que conforman cada equipo se determinarán los respectivos módulos a desarrollar.

7. Cada equipo deberá entregar documentación del diseño y arquitectura de la aplicación.

8. Se pretende que al finalizar un parcial, cada equipo presente el avance del desarrollo del proyecto, así mismo, al finalizar el curso, se realizará una presentación final de la aplicación desarrollada.

UNIDAD I

CONCEPTOS GENERALES SOBRE PARADIGMAS DE PROGRAMACIÓN

1.1. Introducción
    1.1.1. Definición de paradigma
    1.1.2. Lenguajes de programación
1.2. Paradigmas de programación
    1.2.1. Imperativa
    1.2.2. Funcional
    1.2.3. Lógica
    1.2.4. Orientada a objetos
    1.2.5. Orientada a eventos
    1.2.6. Otros paradigmas

Capítulo 1

Conceptos generales sobre paradigmas de programación

1.1. Introducción

La importancia de revisar diferentes tipos de paradigmas de programación es para conocer diferentes formas de abordar la solución a un problema. Cada paradigma posee ventajas y desventajas, así como su facilidad y complejidad en su aplicación. Actualmente el paradigma de programación Orientado a Objetos es el más común, además, permite trabajar con diversos lenguajes de programación, así como Java, Python, C++, C#, entre otros. Los cambios se notan ligeramente en la sintaxis, sin embargo, la lógica es la misma.

En este curso se abordaremos el paradigma de programación Orientado a Objetos con el lenguaje de programación Java. A medida que nos profundicemos en el curso, se revisarán contenidos teóricos y prácticos.

Los paradigmas de programación nos indican las diversas formas que, a lo largo de la evolución de los lenguajes, han sido aceptados como estilos para programar y para resolver los problemas por medio de una computadora.

1.1.1. Definición de paradigma

Thomas Kuhn (1992 - 1996) Es un modelo o patrón aceptado, un arquetipo de investigación que tiene vigencia por determinado tiempo.

Nicola Abbagnano (1901 - 1990) Es un conjunto de creencias y actitudes, como una visión del mundo "compartida" por un grupo de científicos que implica una metodología determinada.

Paradigma de programación

Es un modelo básico de construcción de programas. Un modelo que permite producir programas conforme con unas directrices específicas, tales como diseñar un programa mediante una secuencia de instrucciones que operan sobre unos datos de entrada y producen un resultado de salida (Alonzo, Martínez, Introducción a la ingeniería de software).

1.1.2. Lenguajes de programación

Es un conjunto de reglas, notaciones, símbolos y/o caracteres que permiten a un programador poder expresar el procesmiento de datos y uss estructuras en el computadora.
Cada lenguaje posee su propia sintaxis:

- **Lexico**: conjunto de símbolos que se pueden usar en un lenguaje.
  - **Identificadores:** nombre simbólicos, que se darán a ciertos elementos de programación (nombre de variables, tipos y módulos).
  - **constantes:** datos que no combiarán sus valores a lo largo del programa.
  - **Operadores:** símbolo que representarán operaciones entre variables y constantes.
  - **Instrucciones:** símbolos especiales que representarán estructura de procedimiento, y definición de elementos de programación.
  - **Comentarios:** texto que se usará para documentar los programas.
- **Sintaxis:** consta de unas definiciones, denominadas sintácticas o producciones que especifican la secuencia de símbolos que forman una frase del lenguaje. Estas reglas dicen si una frase está bien escrita o no.
- **Semática:** define el significado de las construcciones sintácticas del lenguaje y de las expresiones y tipos de datos utilizados.

Hay diferentes tipos de lenguajes, así como los compilados y también los interpretados. En la imagen 1.1 se muestran algunos de los lenguajes que son más empleados en la actualidad.

Figura 1.1: Lenguajes de programación (*fuente: https://codigoonclick.com/mejores-lenguajes-programacion-para-2018/*)

1.2. Paradigmas de programación

1.2.1. Imperativa

1.2.2. Funcional

1.2.3. Lógica

1.2.4. Orientada a objetos

1.2.5. Orientada a eventos

1.2.6. Otros paradigmas

Capítulo 2

Paradigma Orientado a Objetos

2.1. Beneficios del paradigma orientado a objetos sobre otros paradigmas

2.2. Elementos primordiales

2.2.1. Clases y objetos

2.2.2. Abstracción

2.2.3. Encapsulamiento

2.2.4. Modularidad

2.2.5. Jerarquía y herencia

2.2.6. Polimorfismo

Capítulo 3

Introducción a la tecnología en un lenguaje Orientado a Objetos

3.1. Introducción

Capítulo 4

Datos, operadores y sentencias de control

4.1. Introducción

Capítulo 5

Métodos y mensajes

5.1. Introducción

Capítulo 6

Constructores

6.1. Introducción

Capítulo 7

Herencia

7.1. Introducción

Capítulo 8

Interfaces gráficas de usuario

8.1. Introducción

Bibliografía

[1] Hahn, J."LaTeX for eveyone". Prentice Hall, New Jersey, 1993.

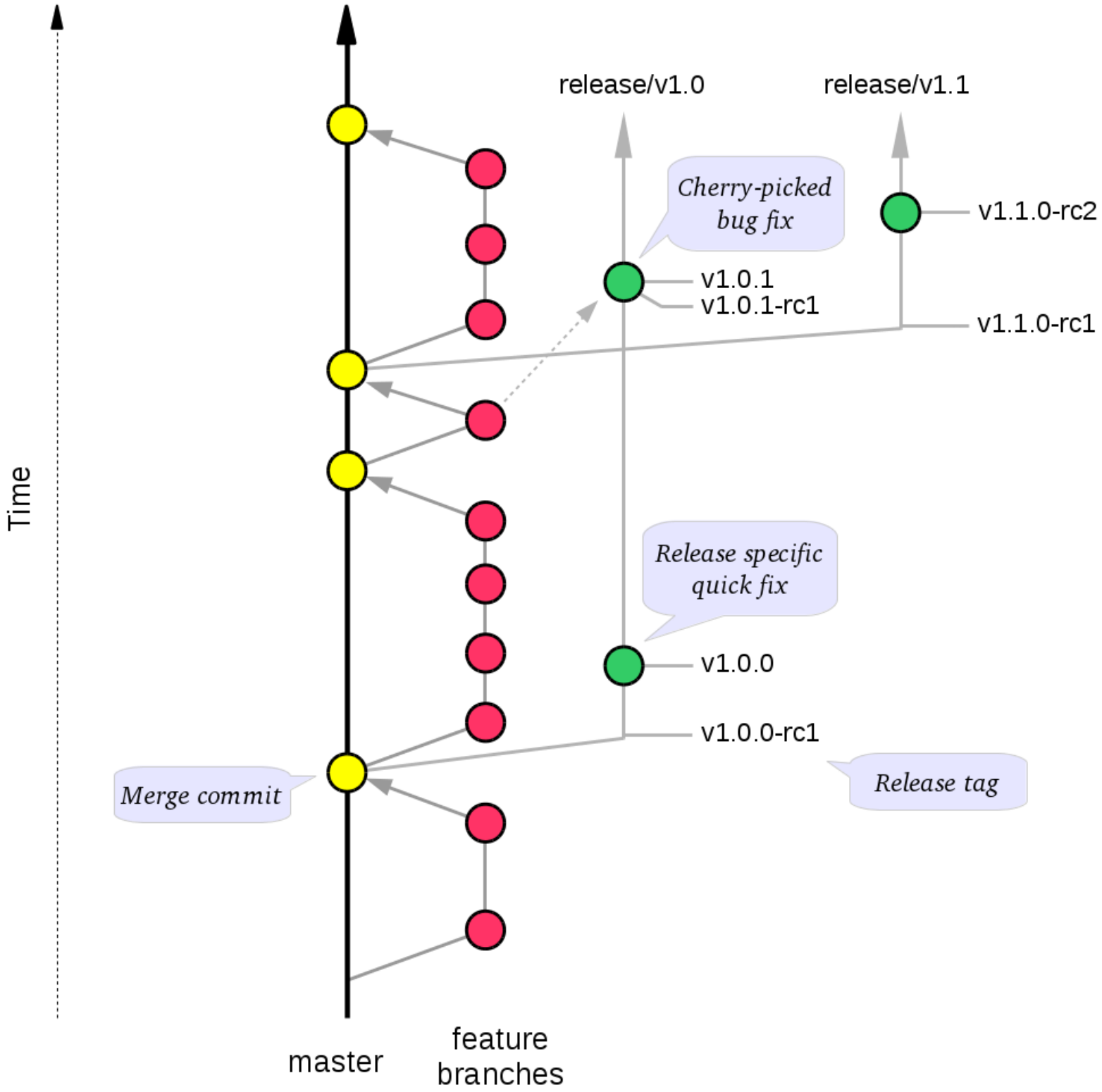
$$Minombrees \tag{8.1}$$
$$rolando \tag{8.2}$$
$$pedro \tag{8.3}$$
$$gabriel \tag{8.4}$$
$$\tag{8.5}$$

Guía.

1. Abra un nuevo archivo en GeoGebra.
2. Oculte los ejes, para esto elija el menú Vista y desmarque la opción Ejes.
3. Elija la herramienta Polígono y construya el triángulo ABC.
4. Utilice la herramienta Punto Medio o Centro para construir los puntos D, E y F que corresponden a los puntos medios de los lados a, b y c en ese orden.


git


App V1
App V2
Clone
Change
Add & Commit
Push


release/v1.0
release/v1.1
Cherry-picked bug fix
v1.1.0-rc2
v1.0.1
v1.0.1-rc1
v1.1.0-rc1
Release specific quick fix
v1.0.0
v1.0.0-rc1
Merge commit
Release tag
Time
master
feature branches

SISTEMA DE CONTROL DE VERSIONES

GIT

https://git-scm.com/

```
$git version      -- Conocer la versión de git
```

```
$git help nombre_comando      -- Ver la ayuda de un comando de git
```

```
$git config - -global user.name “nombre_usuario”      -- Crear usuario para
                                                      -- el manejo de git
```

```
$git config - -global user.name”      -- Visualizar el nombre de usuario
```

```
$git config - -global user.email “mail@mail”      -- Ingresar el correo
                                                  -- electrónico
```

```
$git config - -global user.email                  -- Visualizar el mail
```

```
$git config - -global color.ui true               -- Resaltar colores de los
                                                  -- resultados de los comando git
```

```
$git config - -global - -list                     -- Visualizar todas las
                                                  -- configuraciones realizadas
```

```
$git init                                         -- Inicializar y monitorizar un proyecto con git
                                                  -- (ser recomienda ejecutar una sola vez)

                                                     Nota: genera la rama master
```

```
$git status                                       -- Monitorear el estado del proyecto
```

```
$git add nombreArchivo                            -- Indicar el archivo que se desea
                                                  -- controlar con git
```

```
$git add -A                                       -- Indicar todos los archivos que se desea
                                                  -- controlar con git
```

```
$git commit -m “comentario”                       -- Confirmar los cambios (-m permite
                                                  -- añadir un comentario al commit)
```

```
$git log                                          -- Muestra los commits realizados
```

```
$git log > nombreArchivo.txt                      -- Exportar en un archivo de texto
                                                  -- todos los commits
```

```
$git checkout nombreCommit                        -- Moverse a un commit (versión) del
                                                  -- proyecto (navegar entre commits)
```

```
$git checkout nombreRama          -- Moverse a una rama del proyecto
                                  -- (navegar entre ramas)
```

```
$git reset          -- Similar al comando checkout a diferencia de
                    -- que éste elimina los commits.
```

```
$git reset - -soft nombreCommit   -- Es el reset más simple, y este
                                  -- comando no afecta al proyecto
                                  -- (Working Area). No afecta al
                                  -- código
```

```
$git reset - -mixed nombreCommit  -- Borra el "Strging Area" sin
                                  -- tocar al "Working Area" (no es
                                  -- muy utlizado)
```

```
$git reset - -hard      -- Borra absolutamente todo lo que hay en el
                        -- commit (El más peligroso)
```

```
$git brach          -- Visualiza todas las ramas del proyecto
```

```
$git branch nombreRama          -- Crea una rama del proyecto
```

FUSIONES: es la creación de un nuevo commit uniendo una rama con otra.

```
$git merge nombreRama           -- Fusiona dos ramas, la que se
                                -- encuentra activa actualmente y la
                                -- rama especificada
```

```
$git branch -D nombreRama       -- Elimina la rama especificada
```

```
$git branch -b nombreRama       -- Crea una rama y se cambia
                                -- automáticamente a ella
```


```
$git clone urlProyectoGithub      -- Descargar un proyecto
                                  -- Github sin vinculación
```

```
$git remote add origin urlProyectoGithub   -- Indica a git que
                                           -- el proyecto es
                                           -- el mismo con el
                                           -- que está en el
                                           -- repositorio de
                                           -- Github y que los
                                           -- sincronice
                                           -- (conectar ambos
                                           -- proyectos)
```

```
$git remote -v      -- Comprobar la sincronización de la
                    -- conexión con el proyecto en Github
```

```
$git remote remove origin      -- Elimina el repositorio
                               -- remoto
```

```
$git push origin master    -- Enviar cambios (commits) a github
                           -- (ingresar usuario y contraseña)
```

```
$git commit - -amend –m “Cambiar mensaje”   -- Modificar
                                            -- mensaje del
                                            -- commit
```

```
$git push origin master -f    -- Forzar un push en github
                              -- cuando no hay commits para
                              -- subir, pero se quiere
                              -- modificar un mensaje
```

```
$git push origin nombreRama   -- Subir una rama en github
```

ISSUES: son una forma de continuar, mejorar o solucionar un error en nuestro repositorio (proyecto).

MILESTONES: son grupos de issues que aplican para un proyecto, característica o periodo de tiempo.

LABELS: son una manera de organizar diferentes tipos de problemas.

TAGS ANOTADAS: son almacenadas como objetos completos dentro de la base de Git y contienen más información (Versión al último commit).

```
$git tag -a v1.0 -m "mensaje"
```

```
$git tag -a v1.0 -m "Mensaje" shaCommit
-- Agregar el código SHA y especificar donde se va a aplicar una
-- etiqueta
```

```
$git push origin v1.0          -- Compartir las tags o subirlo a
                               -- github
```

```
$git push origin -tags          -- Subir todos los tags en github
```

```
$git commit -amend -m "Arreglar Mensaje"   -- Hacer cambios o
                                           -- arreglar
                                           -- mensajes
```

git fetch vs gti merge

```
$git branch -a                  -- Ver ramas ocultas
```

```
$git fetch origin               -- Descargar el repositorio
                                -- remoto a la rama oculta
                                -- origin/master
```

```
$git merge origin/master        -- Absorver la rama origin/master
```

Alias en git

```
$git config --global alias.lg "log --oneline --decorate --all --graph"
            Configuración de alias de forma global
```

```
$git lg             -- El alias es lg y se ingresa en la terminal
```

```
$git config --global alias.s "status -s -b"    -- Crear alias para status
```

```
$git s          -- Ejecutar en git el alias de status
```

Toda la información del alias se grava en el global y se puede revisar con el siguiente comando:

```
$git config –global -e
```

No se recomienda visualizar la grabación de los alias desde el archivo, para eso se utiliza el siguiente comando:

```
$git config –global -l
```

MÁS SOBRE EL ARCHIVO .git

Al iniciar git se crea el archivo .git:

Almacena toda la información del repositorio y los archivos que están afuera corresponden al espacio de trabajo

Directorios de .git:

1. .git
1.1. **objects:** almacena los commits y el contenido de los commits (commits, archivos y carpetas)
1.2. **refs:** contiene ramas locales y remotas, y también las etiquetas (heads, remotes, tags)
1.3. **heads:** en esta carpeta se encuentran las ramas locales, que son archivos que tienen el nombre de la rama y como contenido el último commit al que hace esa rama.

1.4. **Rama:** es una no es más que una referencia a un commit (una rama es un archivo alojado en el repositorio).

Nota: el archivo head indica a qué rama, etiqueta o commit corresponde al espacio de trabajo en el que estamos ubicados actualmente y se actualiza cada vez que nos movemos de la rama (con checkout).

1.5. **index:** es el archivo que almacena la lista de cambios que están listos para hacer commit

1.6. **config:** es un archivo de configuración de nuestro repositorio

1.7. **hooks:** nos permite almacenar scripts que se ejecutan en determinados momentos, antes de hacer un push o un commit.

Listar los archivos de la carpeta .git:

```
$ls .git
```