1a GUIA DE POO

(Roberto Tecla)

VÁZQUEZ MORENO MARCOS OSWALDO

2CM3

1.-Diga a que concepto corresponde la definición siguiente: denota las **características esenciales** **que distinguen a un objeto de otra clase de objetos** y provee así una frontera conceptual definida, relativa a la perspectiva del observador.

a) Identidad de un objeto b) Encapsulamiento c) Operación **d) Abstracción**

2.- Diga a que concepto corresponde la definición siguiente: **engloba todas las** (usualmente estáticas) **propiedades del objeto** mas los valores actuales (usualmente dinámicos) de cada una de sus propiedades.

**A) Estado de un objeto** B) Comportamiento de un objeto

C) Identidad de un objeto D) Interfaz de un objeto

3.-Diga a que concepto corresponde la definición siguiente: Es como **actúa** y reacciona **un objeto en términos** de **cambio de estado**.

A) Encapsulamiento **B) Comportamiento de un objeto**

C) Identidad de un objeto D) Interfaz de un objeto

4.-Diga a que concepto corresponde la definición siguiente: es la **propiedad** de un objeto que lo **distingue de todos los otros objetos**

A) Estado de un objeto B) Comportamiento de un objeto

**C) Identidad de un objeto** D) Interfaz de un objeto

5.- Es una **colección de objetos que comparten estructura y comportamiento**

a) herencia **b) clase** c) paquete d) ) interfaz

6.- Diga a que concepto corresponde la definición siguiente: es una **función** o transformación q**ue se puede aplicar o que puede ser aplicada** por los objetos de una clase.

A) Identidad de un objeto B) Encapsulamiento

**C) Operación** C) Abstracción

7.- Diga a que concepto corresponde la definición siguiente: Es una **relación entre clases** en la que **una clase comparte la estructura y/o el comportamiento definido en una o más clases**.

|  |  |
| --- | --- |
| **A) Herencia** | B) Interfaz |
| C) Una clase abstracta | C) Encapsulamiento |

8.-De acuerdo con su **ámbito** que **variables** se pueden usar en mas de un método de la misma clase

**a) Las de instancia** b) Las locales c) Ninguna de las anteriores

9.-De acuerdo con su **ámbito** que **variables** se declaran dentro de un método

a) Las de instancia b) Las de clase **c) Las locales** d) Ninguna de las anteriores

10.-Cual de las siguientes no es una **palabra reservada** en Java?

a) byte c) catch **c) real** d) implements e) static

11.-Los tipos básicos en Java son:

1. String, float, double, integer, short, long, char
2. integer, byte, String, float, short, long
3. **int, byte, double, float, short, long, char, boolean**
4. int, short, long, char, double, float

12.-Cual de las siguientes afirmaciones no es cierta para un constructor ?

|  |  |  |  |
| --- | --- | --- | --- |
| A) Su tipo de retorno es void B) Esta sobrecargado  C) Sirve para inicializar los objetos de la clase D) Se llama como la clase  13.-En Java\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_**solo contiene constantes y métodos abstractos**  A) Una clase abstracta B) Una abstracción  C) Una interfaz D) Una operación  Diga si las afirmaciones siguientes son **verdaderas(**V) **o falsas** (F): | | ( A )  ( C ) | |
| 1.-El **encapsulamiento** permite ocultar los detalles de implementación de los objetos de una clase | | ( V ) | |
| 2.- Si un programa en Java tiene un método **main** podemos asegurar que es una **aplicación** | | ( V ) | |
| 3.-Si un programa en Java tiene un método **main** podemos asegurar que es un **applet** | | ( F ) | |
| 4.-Si un programa en Java tiene como superclase a la clase **Applet** podemos asegurar que es un ( V ) **applet**  5.-El **estado** es lo que permite distinguir a un objeto de otro ( F )  6.-La **herencia** en Java es múltiple ( F )  7.-La **herencia** en Java es simple ( V )  8.-Es la  **herencia** la que permite usar métodos y atributos definidos en una superclase ( V ) | | | |
| 9.-El nombre del **constructor** de los objetos de una clase no tiene que ser el mismo que el nombre de dicha clase | | ( F ) | |
| 10.-El **constructor** de los objetos de una clase tiene como tipo de retorno **void** | | ( F ) | |
| 11.-Si 2 o mas métodos tienen el mismo nombre pero distinta implementación entonces el método esta **sobrecargado** | | ( V ) | |
| 12.-Si hay mas de un constructor en una clase entonces el **constructor** esta **sobrecargado** | | ( V ) | |
| 13.-Una **clase abstracta** no puede tener instancias directas | | ( V ) | |
| 14.-Una **clase abstracta**, en Java, no puede tener variables de instancia | | ( F ) | |
| 15.-En Java si una clase tiene un **método abstracto** entonces es una clase abstracta | | ( V ) | |
| 16.-Un método **abstracto** no tiene implementación | | ( V ) | |
| 17.-Un método **abstracto** no se define es decir solo se declara | | ( V ) | |
| 18.-Una **interfaz**, en Java, solo contiene **constantes y métodos abstractos** | | ( V ) | |
| 19.-Todo lo que esta dentro de una **interfaz** es **publico** | | ( V ) | |
| 20.-Todo lo que esta dentro de una **interfaz** es **privado** | | ( F ) | |
| 21.-Una **interfaz** puede contener métodos abstractos y variables de instancia | | ( V ) | |
| 22.-Una clase no puede **implementar** más de una **interfaz** | | ( F ) | |
| 23.-La clase que **implementa** una interfaz no tiene que implementar todos los métodos contenidos en dicha **interfaz** | | ( V ) | |
| 24.-Una **variable de clase** es compartida por todos los objetos de la clase | | ( V ) | |
| 25.-En Java se pueden **concatenar cadenas** usando el operador + | | ( V ) | |
| 26.-Se usa nombreObjeto.metodo(listaParametrosReales) para llamar un **método de instancia** | | ( V ) | |
| 27.-Se usa nombreClase.metodo(listaParametrosReales) para llamar un **método de instancia** | | ( F ) | |

**Problema 1.** Codificar un applet que tenga dos botones y dos campos de entrada como se muestra en la figura. Si el usuario presiona el primer botón la cantidad en el primer campo de entradas se convertirá a dolares y si se presiona el segundo dicha cantidad se convierte a pesos. El segundo campo se usa para meter el factor de conversión.

|  |  |  |  |  |  |  |  |  |
| --- | --- | --- | --- | --- | --- | --- | --- | --- |
| |  |  |  |  |  |  |  |  | | --- | --- | --- | --- | --- | --- | --- | --- | | |  | | --- | | 0 | | |  |  |  | | --- | --- | --- | | Pesos a Dolares |  | Dolares a Pesos | | |  | | --- | | 15 | | |

import java.awt.\*; import java.awt.event.\*; import java.applet.\*; public class problema1 extends Applet implements ActionListener{

Button p\_d, d\_p;

Label l1, l2;

TextField dinero, factorC; int num=0;

public void init(){ dinero=new TextField(10); p\_d=new Button("Pesos a Dolares"); d\_p=new Button("Dolares a Pesos"); factorC=new TextField(10);

add(dinero); add(p\_d); add(d\_p); add(factorC);

p\_d.addActionListener(this); d\_p.addActionListener(this);

}

public void actionPerformed(ActionEvent e){

Button b=(Button)e.getSource(); String dat = dinero.getText(); int cant = Integer.parseInt(dat); String fac = factorC.getText(); int fact = Integer.parseInt(fac);

if(p\_d==b) num=cant/fact; if(d\_p==b)

num=cant\*fact;

dinero.setText("R: "+num);

}

}

**Problema 2.** Escriba un applet que tenga dos campos de entrada y un botón que al ser presionado muestre en una etiqueta un mensaje que diga si los contenidos de los 2 campos son iguales o no lo son.

import java.awt.\*; import java.awt.event.\*; import java.applet.\*;

public class contenido extends Applet implements ActionListener{

Button mostrar;

Label l;

TextField text1, text2; int num=0;

public void init(){ text1=new TextField(10); text2=new TextField(10); mostrar=new Button("Son iguales?"); l=new Label("\*\*\*\*\*\*\*\*\*");

add(text1); add(text2); add(mostrar); add(l); mostrar.addActionListener(this);

}

public void actionPerformed(ActionEvent e){

Button b=(Button)e.getSource();

/\*String cad1 = text1.getText(); String cad2 = text2.getText();\*/

if(text1.getText().equals(text2.getText())) l.setText("Son iguales");

else

l.setText("NO son iguales");

}

}

**Problema 3.** Escriba un applet que tenga un campo de entrada donde el usuario escribirá una edad y un botón que al ser presionado muestre en una etiqueta el año en que morirá dicha persona. Suponga que la esperanza de vida es de 77 años.

import java.awt.\*; import java.awt.event.\*; import java.applet.\*;

public class edad extends Applet implements ActionListener{

Button morir;

Label l;

TextField edad;

public void init(){

edad=new TextField(5); morir= new Button("Fecha de muerte"); l=new Label("\*\*\*\*\*\*\*\*\*");

add(edad); add(morir); add(l);

morir.addActionListener(this);

}

public void actionPerformed(ActionEvent e){

int num1 = Integer.parseInt(edad.getText());

int year; int num2=77-num1; year=2015+num2;

l.setText("USTED MORIRA EN: " +year);

}

}

**Problema 4.** Codificar un applet que tenga dos campos de entrada y cuatro botones. Los campos de entrada se usaran para escribir dos números, el primer botón, para sumar, el segundo para restar, el tercero para multiplicar y el cuarto para dividir (ver figura). El resultado se imprimirá en la zona de status o una etiqueta (no mostrada aquí).

|  |  |  |  |  |  |  |  |  |  |  |  |
| --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- |
| |  |  |  |  |  |  |  |  |  |  |  | | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | | 0 |  | 0 |  | + |  | - |  | \* |  | / | |

import java.awt.\*; import java.awt.event.\*; import java.applet.\*;

public class calc extends Applet implements ActionListener{

Button sum, rest, div, mult;

Label result; TextField num1, num2;

public void init(){

num1=new TextField(4); num2=new TextField(4); sum= new Button("+"); rest= new Button("-"); div= new Button("/"); mult= new Button("\*"); result=new Label("\*\*\*\*\*\*\*\*\*");

add(num1); add(num2); add(sum); add(rest); add(div); add(mult); add(result);

sum.addActionListener(this); rest.addActionListener(this); div.addActionListener(this);

mult.addActionListener(this);

}

public void actionPerformed(ActionEvent e){ int num1\_int = Integer.parseInt(num1.getText()); int num2\_int = Integer.parseInt(num2.getText());

Button b=(Button)e.getSource();

if(b==sum)

result.setText(""+num1\_int+num2\_int); if(b==rest)

result.setText(""+num1\_int-num2\_int); if(b==mult)

result.setText(""+num1\_int\*num2\_int); if(b==div)

result.setText(""+num1\_int/num2\_int);

}

}

**Problema 5.** Codificar un applet que tenga tres campos de entrada y un botón. Los campos de entrada se usaran para escribir tres números y cuando se presione el botón se mostrara el mayor de los 3 números en una etiqueta.

import java.awt.\*; import java.awt.event.\*; import java.applet.\*;

public class mayor extends Applet implements ActionListener{

Button mayor;

Label result;

TextField num1, num2, num3;

public void init(){

num1=new TextField(4); num2=new TextField(4); num3=new TextField(4);

mayor= new Button("MAYOR"); result=new Label("\*\*\*\*\*\*\*\*\*");

add(num1); add(num2); add(num3); add(mayor); add(result); mayor.addActionListener(this);

}

public void actionPerformed(ActionEvent e){ int num1\_int = Integer.parseInt(num1.getText()); int num2\_int = Integer.parseInt(num2.getText()); int num3\_int = Integer.parseInt(num3.getText());

if(num1\_int>num2\_int) if(num1\_int>num3\_int)

result.setText("EL NUMERO MAYOR ES"+num1\_int); else if(num1\_int==num3\_int)

result.setText("LOS NUMEROS MAYORES SON "+num1\_int+ "y"+num3\_int); else if(num2\_int>num3\_int)

result.setText("EL NUMERO MAYOR ES"+num2\_int); else if(num2\_int==num3\_int)

result.setText("LOS NUMEROS MAYORES SON "+num2\_int+ "y"+num3\_int);

else

result.setText("EL NUMERO MAYOR ES"+num3\_int);

}

}

**Problema 6.** Codificar un applet que tenga tres botones y una etiqueta, y que si se presiona un botón entonces se muestre el texto que esta dentro del botón (Manzana, Limón o Naranja) en la etiqueta.

|  |  |  |  |  |
| --- | --- | --- | --- | --- |
| |  |  |  |  | | --- | --- | --- | --- | | |  | | --- | | Limon | | |  | | --- | | Naranja | |   ManzanaManzana |

import java.awt.\*; import java.awt.event.\*; import java.applet.\*;

public class calc extends Applet implements ActionListener{

Button manzana, naranja, limon;

Label l;

public void init(){

manzana= new Button("manzana"); naranja= new Button("naranja"); limon= new Button("limon"); l=new Label("\*\*\*\*\*\*\*\*\*"); add(manzana); add(naranja); add(limon); add(l);

manzana.addActionListener(this); naranja.addActionListener(this);

limon.addActionListener(this);

}

public void actionPerformed(ActionEvent e){ Button b=(Button)e.getSource(); l.setText(b.getLabel());

}

}

GUIA I 2a parte POO I

(Roberto Tecla)

|  |  |  |
| --- | --- | --- |
| **Foco** |  | **Apagador** |
| estado wattage | Posición |
| encendido()  apagado() | subir() bajar() |

1.-Modelar una clase **Foco** y una clase **Apagador** .

100 W  

public class Foco{

|  |  |  |  |
| --- | --- | --- | --- |
| } | private boolean estado; private int wattage;  publicFoco(){  estado=false;  } public void encendido(){  estado=true;  } public void apagado(){  estado=false;  } | public class Apagador{ | |
|  | private bolean posicion; private Foco foco; public Apagador(Foco foco){ this.foco=foco; posicion=false;  } public void subir(){  posicion=true;  foco.encender();  }  public void bajar(){ |

posicion=false;

foco.apagado();

}

}

2.-Modelar una **clase abstracta** Mascota que tenga un método **hablar** abstracto y codificar **subclases** de dicha clase por ejemplo perro, gato , canario, etc.



Miau

Woof

Tweet

|  |  |  |  |  |  |  |  |  |  |  |  |  |  |
| --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- |
| |  | | --- | | **Gato** | |  | | hablar() | | | |  |  |  | | --- | --- | --- | | **Perro** |  | **Canario** | |  |  | | hablar() | hablar() | | |
| public class Gato extends Mascota{ public void hablar(){ System.out.printl("Miau");  }  } | | public class Perro extends Mascota{ public void hablar(){ System.out.printl("Woof");  }  } | | public class Canario extends Mascota{ public void hablar(){ System.out.printl("Tweet");  }  } | |

3.-Modelar una clase **Recipiente**.

Gasolina

3L 2L Petroleo Agua

22L

|  |
| --- |
| Recipiente |
| capacidad contenido contiene |
| vaciar()  llenar() |

public class Recipiente { private int capacidad; private string contenido; private int contiene; public Recipiente(){

capacidad = 22; contenido = “Gasolina”;

contiene = 0;

}

public void vaciar(){

contiene = 0;

}

public void llenar(int valorAllenar){ if(valorAllenar > capacidad){

System.out.println(“Se va a tirar ” + contenido + “ porque estamos llenando el recipiente de más, el recipiente es de ” + capacidad + “ Litros”);

}else{

contiene = valorAllenar;

}

}

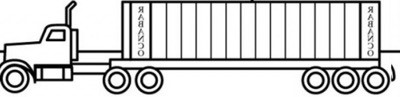
}

Nota: la variable contiene es la cantidad de contenido que hay en el recipiente.

4.-Modelar una clase **Remolque** y una clase **Trailer**.

Trailer Remolque

!



|  |  |
| --- | --- |
| Trailer | Remolque |
| llantas  velocidad velocidadMax | capacidad peso  longitud trailer |
| acelerar() frenar() unirse() | conectar() |
| public class Trailer {  private int llantas; private int velocidad;  public Trailer(int llantas, int velocidadMax){ this.llantas = llantas;  this.velocidadMax = velocidadMax;  }    public void acelerar(){ int velocidadActual = velocidad  for(int i = velocidad; i < velocidadMax; i++){  velocidad=+i;  }  }  public void frenar(){ int velocidadActual = velocidad for(int i = velocidadActual; i >= 0; i-- )  {  velocidad = i;  }  }  public void unirse(){  System.out.println(“Se ha conectado al trailer un remolque”);  }    } | | public class Remolque {  public Trailer trailerAUsar;  private int capacidad; private int peso; private int longitud;  public Remolque(){  trailerAUsar = new Trailer(12,80);  capacidad = 30; peso = 80;  longitud = 15;  }    public void conectar(){  trailerAUsar.unirse();  }    } | |

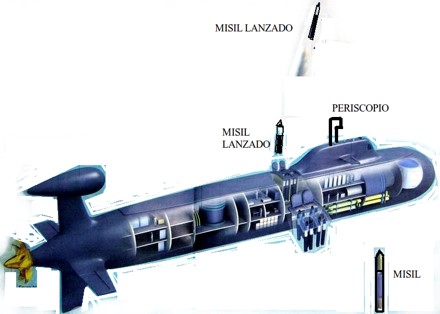
5.-Modelar una clase **Asiento** y una clase **Camión** (de pasajeros).

Asiento



|  |  |  |
| --- | --- | --- |
| Asiento | Camión | |
| posiciónX posiciónY estado | asientos totalPasajeros | |
| sentarse() | llenar() | |
| public class Asiento {  private int posicionX; private int posicionY;  private string estado;  public Asiento(int x, int y){  posicionX = x; posicionY = y;  estado = “vacio”;  }  public void sentarse(string estado){  this.estado = estado;  }    } | | public class Camion { private Asiento asientos[2][14];  private int totalPasajeros; public Camion(){  totalPasajeros =28;  for(int i = 0; i < 2; i++){  for(int k = 0 ; k < 14 ; k++){  asientos[i][k] = new Asiento(i,k);  }  }  }    public void llenar(){ for(int i = 0; i < 2; i++){  for(int k = 0 ; k < 14 ; k++){  asientos[i][k].sentarse(“ocupado”);  }  }  }  } | |

6.-Modelar las siguientes clases **Submarino, Periscopio y Misil.** Considere que un misil tiene que ser armado (o activado) antes de lanzarse**.**

!

|  |  |  |  |
| --- | --- | --- | --- |
| Submarino | Periscopio | | Misil |
| Periscopio  Misil velocidad | activo | | objetivo  motor |
| moverse() | mirar() | | lanzar() |
| public class Submarino { private Misil mMisil; private Periscopio pPeris;  private int velocidad;  public Submarino(int velocidad){  mMisil = new Misil(“motor de  propulsión”, “enemigo”);  pPeris = new Periscopio();  this.velocidad = velocidad;  }  public void moverse(){  System.out.println(“nos estamos moviendo); mMisil.lanzar():  }  } | | public class Periscopio {  private boolean activo; public Periscopio(){  } public class Periscopio {  private boolean activo; public Periscopio(){  activo = false;  }  public void mirar(){  activo = true;  }  }  public class Misil { private string objetivo; private string motor;  public Misil(string motor, string objetivo){ this.objetivo = objetivo;  this.motor = motor;  }  public void lanzar(){  System.out.println(“se ha lanzado el misil a ” + objetivo);  }  } | | |

|  |  |
| --- | --- |
| Camara | Rollo |
| megaPixeles  rollo | tamaño |
| tomarFoto() | revelar() |

7.-Modelar una clase **Cámara** y una clase **Rollo**. Considere que los rollos pueden ser para 12 o 24 o 36 fotos.

|  |  |
| --- | --- |
| public class Camara { private int megaPixeles;  private Rollo rolloCamara;  public Camara(int x){  megaPixeles = x;  rolloCamara = new Rollo(24);  }  public void tomarFoto(){  System.out.println(“se tomo una foto”);  }    } | public class Rollo {  private int tamaño; public Rollo(int tamaño){ this.tamaño = tamaño  }    public void revelar(){  System.out.println(“se revelaron todas las fotos”);  }  } |

**Importante**

En cada modelado identificar el conjunto de operaciones y la representación. También hay que encontrar como se relacionan las clases.

Para cada Modelado además escribir el código en Java

-Definición de clase

-Declarar las variables de instancia (en el problema 5 declarar un arreglo)

-Constructor (en el problema 5 inicializar el arreglo)

-Codificar los métodos que sean convenientes