**GUÍA DE PASOS DE ATRIBUTOS Y MÉTODOS**

|  |  |
| --- | --- |
| **OBJETIVO DE APRENDIZAJE** | **OA 4**. Construir aplicaciones computacionales basadas en programación orientada a objetos, de manera de cumplir con las exigencias técnicas y de los usuarios. |
| **APRENDIZAJE ESPERADO** | **AE 5.** Construye programas, utilizando los componentes del paradigma orientado a objetos, a partir del análisis de problemáticas y requerimientos, modelando el comportamiento del sistema.  **AE 1.** Construye unidades de prueba para verificar el correcto funcionamiento de la codificación realizada, de acuerdo a exigencias técnicas de confiabilidad.  **AE 2.** Detecta y corrige errores de codificación, analizando el comportamiento del código de programación, de acuerdo a especificaciones y manuales de referencia. |
| **CRITERIOS DE EVALUACIÓN** | **5.1.** Construye los componentes a utilizar en el programa a partir del análisis de requerimientos para resolución de problema, resguardando los principios de POO.  **1.1.** Realiza pruebas para detectar problemas previos a la codificación de una unidad de software con una herramienta de software disponible en el mercado.  **1.2.** Construye una unidad de prueba para detectar excepciones en un entorno de lenguaje con una herramienta compatible que permita desarrollar, configurar e Implementar aplicaciones en función del lenguaje utilizado.  **1.3.** Construye una unidad de prueba para detectar casos límites, en un entorno de lenguaje, con una herramienta compatible, propia del software utilizado como marco de trabajo.  **2.1.** Utiliza la herramienta de depuración de un entorno de desarrollo, para revisar la codificación de clases programadas.  **2.2.** Revisa y corrige las variables, del programa desarrollado para la solución de un caso, indagando en las funciones internas de acuerdo a protocolos de revisión.  **2.3.** Revisa las variables de una función de clases programadas, utilizando opciones del lenguaje empleado, para asegurar su correcto funcionamiento. |
| **OBJETIVOS DE APRENDIZAJE GENÉRICOS** | **A – B – C – G - H** |

**ATRIBUTOS**

Los atributos, también llamados datos o variables miembro, describen el estado del objeto. Este componente nos permite diferenciar las características de cada objeto y consta de tres partes:

1. El tipo de dato,
2. Un identificador o nombre de la variable y
3. Un valor asociado.

|  |
| --- |
| public class Bebe{  *//atributos de la clase Bebé*  String nombre;  double peso;  boolean esMujer;  *//Tipo de dato, identificador y valor asociado.*  int numPopo = 0; } |

Cada vez que una clase sea instanciada, el sistema crea una copia de todas las variables que están vinculadas a dicha clase, haciéndolas propias del objeto. Es necesario entregar un valor a dichas variables para que el objeto defina sus características.

Para acceder al atributo de un objeto, en Java se utiliza el operador punto como se muestra a continuación:

System.out.println("Mi nombre es " + chucky.nombre);

**MÉTODOS**

Los métodos determinan el comportamiento de la clase y sus instancias. En el momento de la declaración del método, es necesario indicar qué tipo de parámetro retorna, o void en el caso que no devuelva nada. En Java, para que un objeto ejecute alguno de sus métodos se utiliza el operador punto.

chucky.popo();

En algunos casos el método requiere de mayor información para saber exactamente lo que tiene que hacer. Esa información se pasa a través de parámetros, y es necesario especificar el tipo de dato e identificador de cada uno de ellos. Si un método no posee parámetros, el paréntesis queda vacío. Los parámetros son variables locales a los métodos y existen sólo en el interior de la función. En términos generales la declaración de un método se hace de la siguiente forma:

tipoDato nombreDelMétodo (tipoDato parámetro1, tipoDato parámetro2, ... )

También existe la posibilidad de usar el mismo nombre del método, pero con diferentes argumentos y/o diferentes valores de retorno. A esto se le llama sobrecarga de métodos y hacen más cómoda la implementación de un objeto.

**POR EJEMPLO**

*//Saluda por defecto.*  
void saludo();   
*//Saluda indicando su nombre.*  
void saludo(String nombre);  
*//Saluda indicando su nombre y sexo.*  
void saludo(String nombre, boolean esMujer)  
*//Saluda a otro bebe indicando su nombre y sexo.*  
void saludo(String nombre, boolean esMujer, Bebe otroBebe)

**CONSTRUCTORES**

Un constructor es un método especial que tiene como objetivo inicializar los atributos de un nuevo objeto y se ejecuta al momento de crear una instancia de la clase. Dependiendo del número y tipos de los argumentos, se llama al constructor correspondiente. Si no se ha escrito un constructor de la clase, el compilador proporciona un constructor por defecto vacío, iniciando los atributos por defecto.

Las reglas del constructor son las siguientes:

1. El constructor tiene el mismo nombre que la clase.
2. Puede tener cero o más argumentos.
3. No devuelve ningún valor (ni siquiera *void*).
4. Toda clase debe tener al menos un constructor.

Usando la clase Bebe, su constructor puede ser de esta forma:

public Bebe(String name, boolean mujer, double weight, double db) {  
 nombre = name;  
 esMujer = mujer;  
 peso = weight;  
 decibeles = db;  
 }

**De esta manera, en el método de la aplicación, se ingresan los valores correspondientes a los atributos definidos en el constructor. Para el caso mostrado a continuación, el objeto “chucky” tiene como nombre “chucky”, es hombre, pesa 3.6 kg y ¡llora emitiendo 100 decibeles!**

Bebe chucky = new Bebe("chuky", false, 3.6, 100);

Una clase puede tener varios constructores y dependiendo del contexto de uso son invocados.

A modo de ejemplo, el siguiente código recapitula lo aprendido hasta el momento:

public class Bebe  
{  
 *//Atributos de la Clase Bebe*  
 public String nombre;  
 boolean esMujer;  
 double peso;  
 double decibeles;  
 int numPopo = 0;  
  
 *//Constructor de la clase Bebe*  
 public Bebe(String name, boolean mujer, double weight) {  
 nombre = name;  
 esMujer = mujer;  
 peso = weight;  
 }  
  
 *//Métodos de la Clase Bebe*  
 void popo() {  
 numPopo += 1;  
 System.out.println("¡Que alguien me cambie el pañal!");  
 }  
  
 public static void main(String[] args) {  
 *//Creando una instancia de Bebe*  
 Bebe chucky = new Bebe("Chucky", false, 3.8);

*//Accediendo a los atributos y métodos de chucky*  
 System.out.println(chucky.nombre);  
 chucky.popo();  
 }  
}

**BUENAS PRÁCTICAS**

1. El nombre del atributo debe ser representativo, de manera tal que le diga al lector lo que representa. Cada palabra interna debe ir en mayúscula. Ejemplo: **promedioParcial, nombreMascota, rut**.
2. Los métodos deben ser verbos en minúscula, con cada palabra interna en mayúscula. Ejemplo: **retirarDinero, correr, correrRapido**.

**RESUELVAN EL SIGUIENTE RETO**

* + - 1. **¿Cuál es la idea general del Reto?**

**R:** Criptomonedas.

* + - 1. **¿Cuál es nuestra pregunta esencial para responder al reto?**

**R:** ¿Cómo funcionan las transferencias de dinero en Internet?

* + - 1. **¿Cuál es nuestro reto o desafío?**

**R:** Crear una clase, siendo programadores de la criptomoneda CryptoPudú.

* + - 1. **¿Qué preguntas, actividades y recursos necesitamos para resolver el reto?**

**R:** Realicen preguntas entre ustedes que les ayude a resolver el reto y generen actividades y/o recursos para representar el conocimiento necesario para desarrollar exitosamente la solución. Pueden apoyarse de la primera parte de esta guía así como también de estas instrucciones:

1. Reúnanse en grupos de pares (dúos o parejas) y desarrollen el siguiente desafío.

Ustedes son un dúo de programadores de la criptomoneda **CryptoPudú**, una divisa electrónica chilena y tienen como objetivo crear la siguiente clase.

1. Clase Persona con las siguientes características:

* **Atributos:** nombre, edad, rut, sexo, PudúCoins.
* **Constructor** con parámetros.
* **Métodos:** abonarDinero, retirarDinero, mostrarDatos, esVip (Si la persona tiene más de 10.000.000 de PudúCoins).

1. **Bonus:** ¿Qué pasa si la persona retira más dinero del que tiene? Realizar validación.
2. Una vez finalizada la Actividad 8, incorpora las pruebas unitarias necesarias para el aseguramiento de la calidad del programa realizado.
   * + 1. **¿Cuáles son las posibles soluciones?**

**R:** Luego de analizar y seleccionar atributos, constructor y métodos, detallen cuáles son las posibles soluciones para el reto encomendado y presenten las condiciones del entregable. A continuación se detallan las instrucciones:

1. Construir un método que deposite el dinero de una persona a otra, considerando la validación anterior.
2. Entregable





1. Consideraciones





* + - 1. **¿Cuáles son los mecanismos de validación de lo efectuado?**

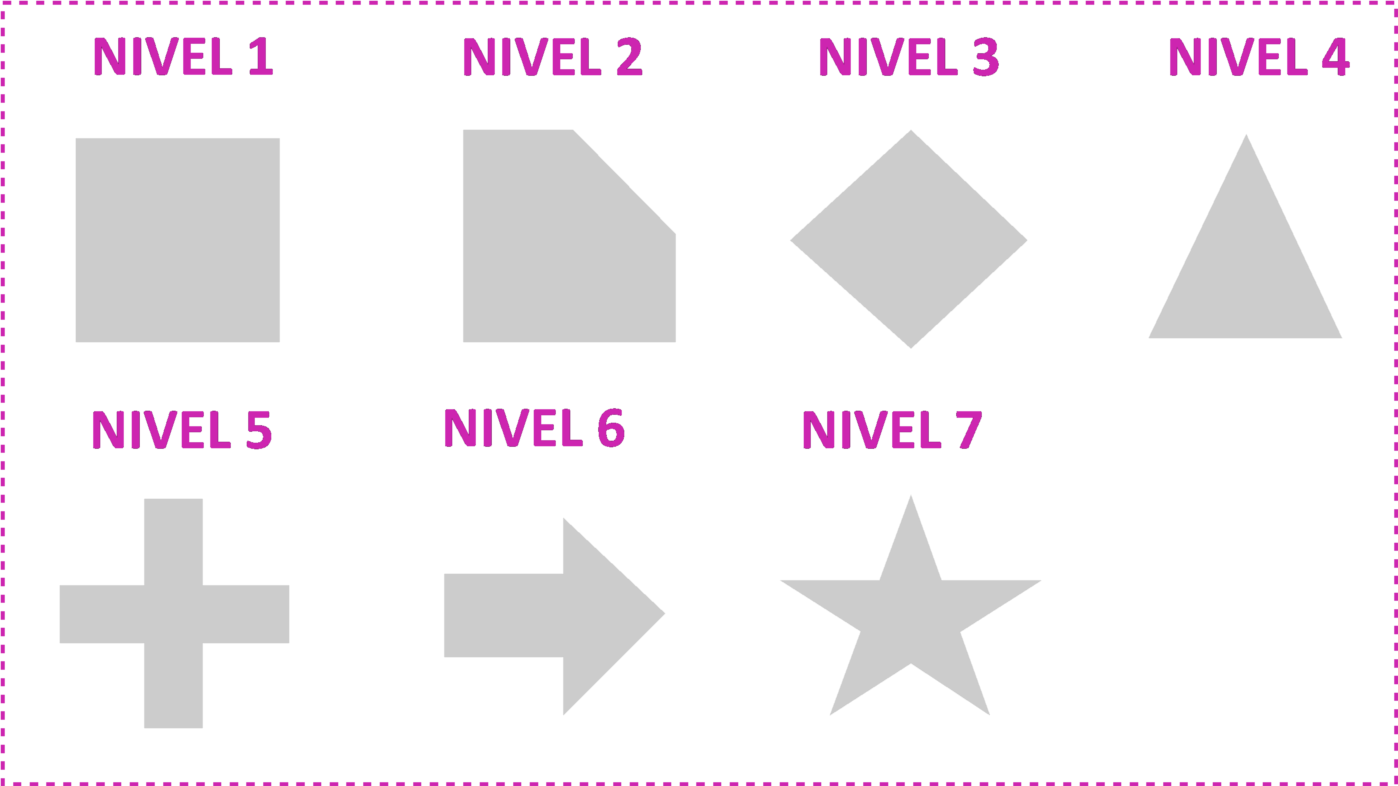
**R:** Creen un método que les permita validar la solución encontrada. En esta etapa trabajen en conjunto con el/la docente. Pueden utilizar los instrumentos de evaluación disponibles y autoevalúen sus desempeños como una oportunidad mejora y autorregulación de los aprendizajes.

* + - 1. **¿Cómo documentamos o publicamos nuestra solución?**

**R:** Para presentar o publicar la solución al reto deben subirlo a **GitHub.**

* **REALICEN EL SIGUIENTE DESAFÍO considerando los pasos anteriores.**

**En los mismos grupos, usen la clase tortuga. Esta define atributos para crear tortugas gráficas como su color y posición, y métodos para lograr que se muevan.** Deberán crear las formas que aparecen a continuación:



Fuente: Elaboración propia.

* 1. Los movimientos que puede realizar la tortuga están detallados a continuación:
* **forward(); //** la tortuga se mueve hacia adelante la distancia de 100 píxeles.
* **forward(x); //** la tortuga se mueve hacia adelante la distancia de x píxeles
* **backward(); //** la tortuga se mueve hacia atrás la distancia de 100 píxeles.
* **backward(x); //** la tortuga se mueve hacia adelante la distancia de x píxeles.
* **turnRight(); //** la tortuga gira 90 grados a la derecha.
* **turnLeft(); //** la tortuga gira 90 grados a la izquierda.
* **turn(x); //** la tortuga rota x grados. Valores positivos rota hacia la derecha, valores negativos, rota a la izquierda.
  1. En el siguiente link podrán programar las tortugas: [Click Aquí.](https://repl.it/@BerylHoffman/Java-Swing-Turtle#Main.java)
  2. Para mayor información de la actividad, revisar Presentación de Atributos y métodos.
  3. Entregables



* 1. Consideraciones











**RECURSOS**

1. Programación ATS. Concepto y creación de métodos: <https://www.youtube.com/watch?v=hDPSRbXBi8U>
2. Programación ATS. Curso completo de programación en Java: <https://www.youtube.com/watch?v=2ZXiuh0rg3M&list=PLWtYZ2ejMVJkjOuTCzIk61j7XKfpIR74K>
3. TutorialesYa. Curso completo de programación en java: <http://www.tutorialesprogramacionya.com/javaya/>
4. La Geekipedia de Ernesto. Curso de Java desde cero:  
   <https://www.youtube.com/watch?v=L1oMLsiMusQ&list=PLyvsggKtwbLX9LrDnl1-K6QtYo7m0yXWB>
5. Java desde cero:   
   <https://javadesdecero.es/>
6. w3schools:  
   <https://www.w3schools.com/java/default.asp>
7. CodingBat:  
   <https://codingbat.com/java>
8. Learn Java:  
   <https://www.learnjavaonline.org/>
9. CodeGym:  
   <https://codegym.cc/>
10. Programming Skills:  
    <http://www.pskills.org/corejava.jsp>
11. Tutorials Point:  
    <https://www.tutorialspoint.com/java/java_generics.htm>
12. Java Tests:  
    <https://www.studytonight.com/tests/?subject=java>
13. JavatPoint:  
    <https://www.javatpoint.com/core-java-quiz>
14. Recopilación de ejercicios Java POO  
    <https://www.academia.edu/36182478/Recopilaci%C3%B3n_de_Ejercicios_Java_POO_Programaci%C3%B3n_Orientada_a_Objetos>
15. Curso de programación orientada a objetos: <https://www.curriculumnacional.cl/614/w3-article-209545.html>
16. Curso de desarrollo de videojuegos: <https://www.curriculumnacional.cl/614/w3-article-209538.html>
17. Curso de desarrollo de aplicaciones móviles: <https://www.curriculumnacional.cl/614/w3-article-209536.html>
18. makigas. Tutorial de GIT: <https://www.youtube.com/watch?v=jSJ8xhKtfP4&list=PLTd5ehIj0goMCnj6V5NdzSIHBgrIXckGU>

**REFERENCIAS**

1. MIT. (2010). Introduction to Programming in Java, Lecture 4: Classes and Objects. Recuperado 20 de agosto de 2020, de <https://ocw.mit.edu/courses/electrical-engineering-and-computer-science/6-092-introduction-to-programming-in-java-january-iap-2010/lecture-notes/MIT6_092IAP10_lec04.pdf>
2. EHU. (n.f). Conceptos básicos de la programación orientada a objetos. Recuperado 20 de Agosto de 2020, de <http://www.sc.ehu.es/sbweb/fisica/cursoJava/fundamentos/clases1/clases.htm>