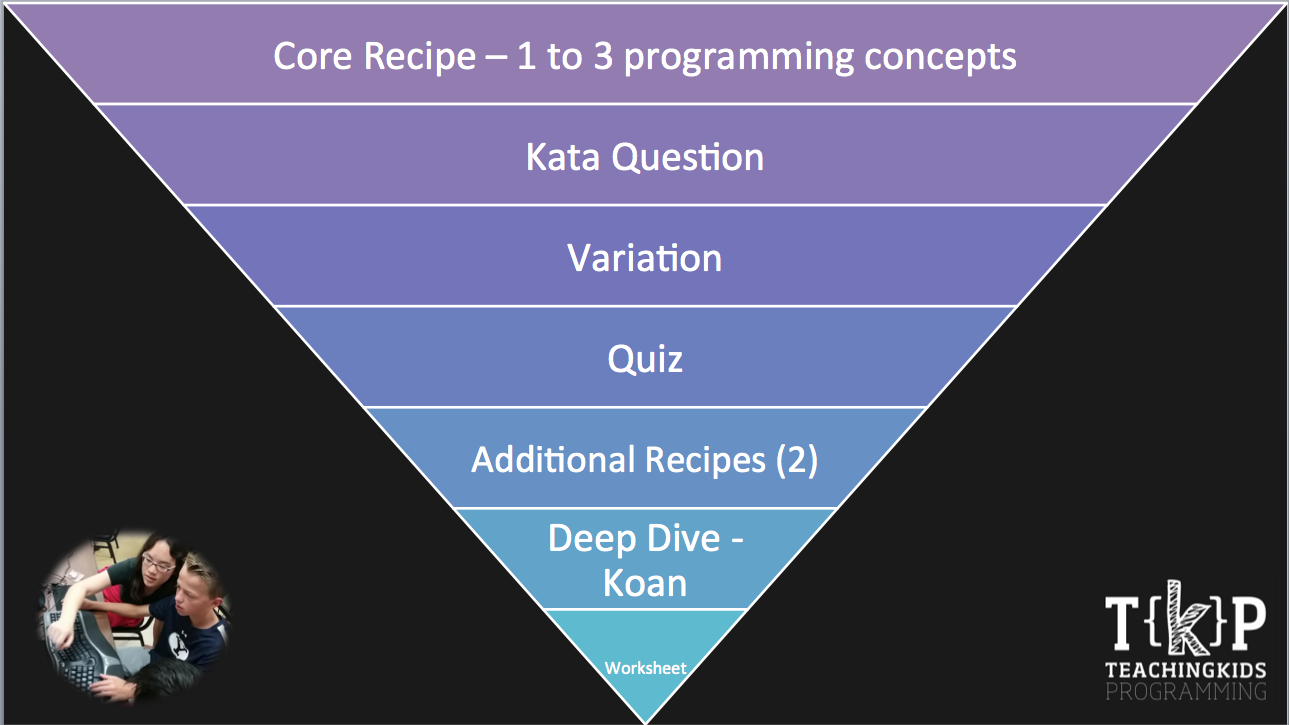
**TKP - Información sobre el Diseño del Programa**

****

**Temas Cubiertos:**

* 5 Pasos para Enseñar el Material Didáctico de TKP
* Sobre el Método Intencional de TKP
* ¿Cuáles son las Partes de cada curso de TKP?
* ¿Cómo TKP usa las Prácticas y Términos de Técnicas Ágiles?
* ¿Por qué Programar en Pareja?
* ¿Por qué Programar en Grupo?
* Tips para la Administración del Salón de Clases para TKP y la Utilidad Virtual de Proctor de TKP
* Guía para el Material Didáctico de el Java de TKP y el Código Fuente de Github
* Guía para Crear Recetas de TKP
* Material Didáctico de TKP para otros Lenguajes de Computadora - SmallBasic, C#, T-SQL y más
* Sobre el Java de TKP y el Examen APJava

## **5 Pasos para Enseñar el Material Didáctico de TKP**

El Material Didáctico de TKP está diseñado como una guía para profesores por parte de profesionales de enseñanza en K-12 (dirigido principalmente para las escuelas). Los profesores de TKP no necesitan tener experiencia programando en Java, pero ellos deben de conocer que se espera que aprendan este lenguaje mientras completan todo el programa por ellos mismos, antes de que empiecen a enseñar. El material didáctico de TKP de Java es un conjunto de bibliotecas personalizadas (código) de Java, planes de lecciones para los profesores (*GoogleDocs*) y videos en YouTube.

*Nota: Si está familiarizado con la programación en Java, puede empezar descargando directamente el código de Java de TKP y el editor Eclipse personalizado desde Github (Las instrucciones de Instalación detalladas para computadoras de escritorio se encuentran en Github****)*** [***aquí***](https://github.com/swandarina/TKP.Java)***.***

A continuación están los pasos que le preparará para enseñar el material didáctico de TKP:

1. **Lea sobre los Métodos de Diseño Instruccional de TKP.** La información principal está en esta página y también encontrará enlaces a lecturas con más detalles y antecedentes.
2. **Haga las recetas del curso que haya escogido enseñar como lo haría un estudiante.** Esto significa que debe de escribir el código de las lecciones en un editor (Eclipse) usted mismo.
3. **Practique enseñando a uno o más estudiantes (niños o adultos).** Especialmente si es nuevo programando, los profesores de TKP se han sentido más cómodos enseñando a su “clase principal”, después de que han practicado con pequeños grupos.
4. **Revise los recursos disponibles para cada capítulo de TKP.**  Los recursos incluyen explicación escrita, principalmente línea por línea para cada curso en GoogleDocs, también los archivos de respuestas. Además, algunas lecciones tienen videos en YouTube con las respuestas.
5. **Use los Recursos de TKP.** Nuestro recurso actual es la Página Web de TKP / Proctor, la cual le permitirá ver el trabajo de los estudiantes desde una ventana de navegación. Más recursos están siendo planificados.

**Sobre el Método Intencional de TKP**

Hemos desarrollado un nuevo método para enseñar a los niños a programar, el cual llamamos **método intencional**. Nuestro trabajo consiste en personalizar el material didáctico y enseñar técnicas.

 **IDEA PRINCIPAL:** *El Método Intencional es la enseñanza guiada a los niños (organizados en parejas) para que traduzcan los comentarios en Español/Inglés a código ejecutable. El lenguaje de programación que usamos es Java.*

*Hemos creado pequeños experimentos usando el Método Intencional en otros lenguajes. Como por ejemplo, Microsoft SmallBasic, T-SQL y Microsoft Kodu (programación visual). También estamos trabajando en algunos proyectos de IoT (el internet de las cosas)..*

Estos son algunos conceptos fundamentales que hemos procurado seguir al escribir y enseñar todos nuestros capítulos de TKP. Estos conceptos incluyen los siguiente:

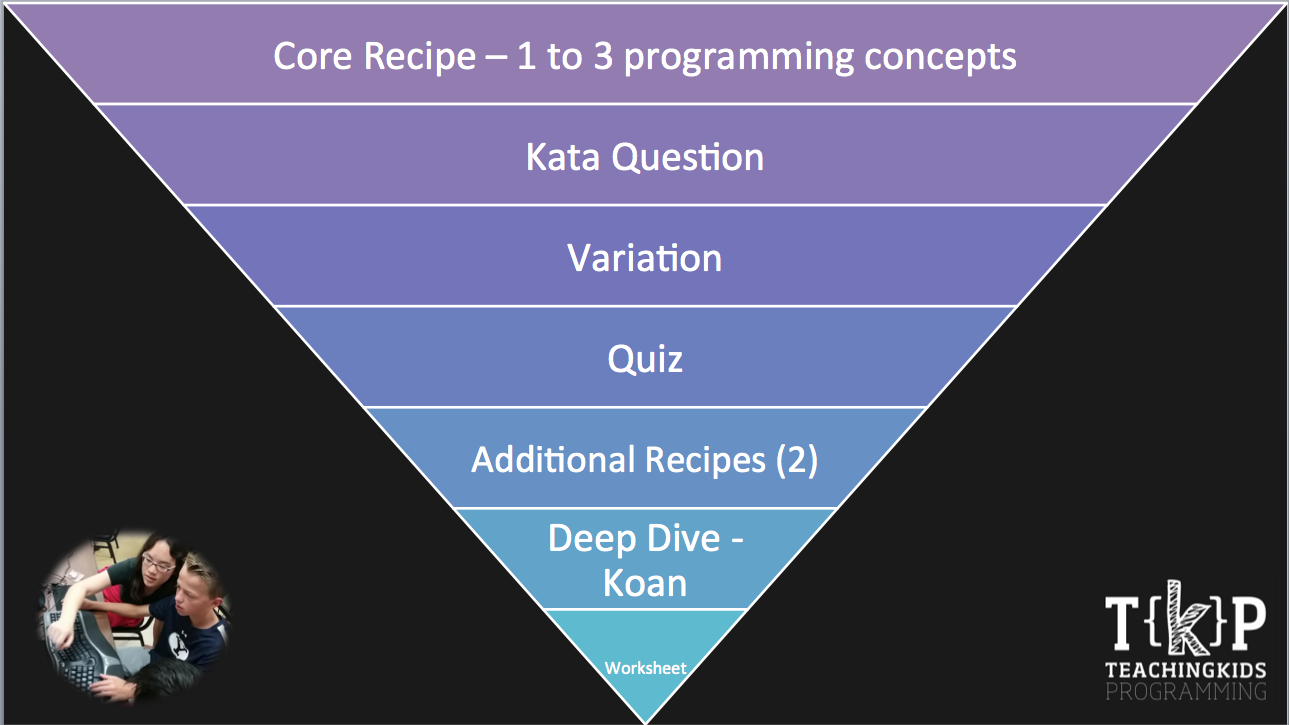
1) **Una línea de Español/Inglés = Una línea de Código** - la exploración del API a través de Intellisense (*aplicación para autocompletar códigos)* es un aspecto clave aquí. Hemos puesto mucha atención en “balancear” estos dos aspectos: nuestro API de Java de TPK para “cubrir conceptos principales y complejos de Java”; y también en la elaboración de un conjunto de comentarios en un Español/Inglés estándar, que representen conceptos comunes en Java. Por ejemplo, la orden: “Establece el valor de la longitud actual a ‘40’”, usa deliberadamente “longitud actual” para indicar que el estudiante debe primero buscar vía Intellisense, para ver si una variable llamada “longitud” está dentro de sus opciones, y si no, debería de crear una nueva variable de ese nombre.

2) **Escriba el código de alguna(s) línea(s) y luego ejecute su programa para verificar** que su traducción fue la correcta. Las lecciones fueron elaboradas para que sean completadas en un orden específico (hay números al final de cada instrucción escrita en Español/Inglés para indicar el orden de traducción y ejecución. Existen instrucciones complejas que estarán divididas en varias líneas y que serán enumeradas usando subíndices, como por ejemplo: 1.1, 1.2, etc..).

3) Después de la verificación, elimine esa línea de los comentarios en Español/Inglés. Nuestras lecciones y el API están elaborados de tal forma para que sean descubiertos por los estudiantes (con la dirección del profesor). Los profesores deberían alentar a los alumnos a usar Intellisense (Ctrl+Espacio, or Cmd+Espacio) y a leer la lista desplegable de los ítems disponibles para ver cuales se relacionan con el Español/Inglés. Los profesores **No**  deben escribir el código de Java antes de que los estudiantes lo hagan. Los profesores **No** deben permitir que los estudiantes copien el código de Java desde el pizarrón, en vez de ello, deben de pedir a los estudiantes a leer los comentarios en Español/Inglés en voz alta si los estudiantes se quedaron atascados y no pueden avanzar. Los profesores deberían dirigir a los estudiantes a leer/usar los ejemplos en el Javadoc.**Enseñar sobre la exploración del API y el uso de las herramientas es un aspecto fundamental para enseñar el Java de TKP.**

 Nota: Solo para la preparación de los profesores, hemos agregado las “respuestas” a todas las lecciones del curso en el siguiente repositorio de GitHub en la siguiente [ubicación](https://github.com/swandarina/TKP-WANDARINA-Recursos_en_Java/tree/master/src/main/java/org/teachingkidsprogramming/recipes/completed).

4) **Dominando conceptos a través de métodos múltiples** de uso - hay entre 6 a 8 lecciones por curso, que cubren los conceptos particulares de programación siendo enseñados de maneras duplicadas e incrementales. Hemos diseñado un material didáctico que se nivela y desarrolla cuidadosamente para que los estudiantes aprendan y dominen entre 1 o 2 nuevos conceptos computacionales en cada etapa. En la parte inferior está un gráfico que muestra las partes (tipos de lección) en cada curso de el Java de TKP en orden sugerido para la enseñanza; por ejemplo, el ítem de nivel superior (lecciones/recetas) están diseñados para ser diseñados al inicio de la lección en cada curso, etc.



Aquí hay un video de bienvenida de 20 minutos para los profesores de k-12 en cómo usar los recursos del material didáctico de el Java de TKP.

[](https://www.youtube.com/watch?v=CNiPrGuR_7U)

## **¿Cuáles son las Partes de cada Curso de TKP?**

**1) Receta/Lección** - Esta es una maestro-guía receta de código que enseña de 1 a 3 conceptos de programación, tales como “¿qué es una clase o un método?”. Se espera que los estudiantes programen en parejas, de acuerdo a las direcciones de los profesores, traduciendo una línea del Español/Inglés en una línea de código, y después ejecutando el programa para verificar la traducción correcta después de cada línea traducida.

Las lecciones están escritas para ser traducidas y ejecutadas en un orden ESPECÍFICO. Este orden es indicado por los números de las líneas que están al final de cada línea de comentarios en Español/Inglés. Las razones por las cuales se ordenó, son las siguientes:

* Presenta los conceptos uno por uno (algunas veces directamente reforzando el aprendizaje con repetición)
* Ayuda a adaptarse poco a poco del lenguaje de comentarios (meta) en Español/Inglés al lenguaje de la biblioteca de TKP Java (término). En otras palabras empezamos por relación directa, por ejemplo, ‘La Presentación de la Tortuga’ se convierte en ‘Tortuga.presentación();’ o ‘Tortoise.show();’ y y avanzar hacia una traducción menos directa, por ejemplo, ‘Haz a la tortuga mover tan rápido posible’, se convierte en ‘Tortuga.establecerVelocidad(10);’ o ‘Tortoise.setSpeed(10);’. Hemos incluido Javadocs personalizables con ejemplos para dar soporte a la habilidad de los estudiantes para descubrir la traducción correcta.

En algunas lecciones también hemos introducido errores de tiempo de ejecución y lanzado mensajes de error personalizables, para presionar más en el uso/lectura del documento mientras se está programando.

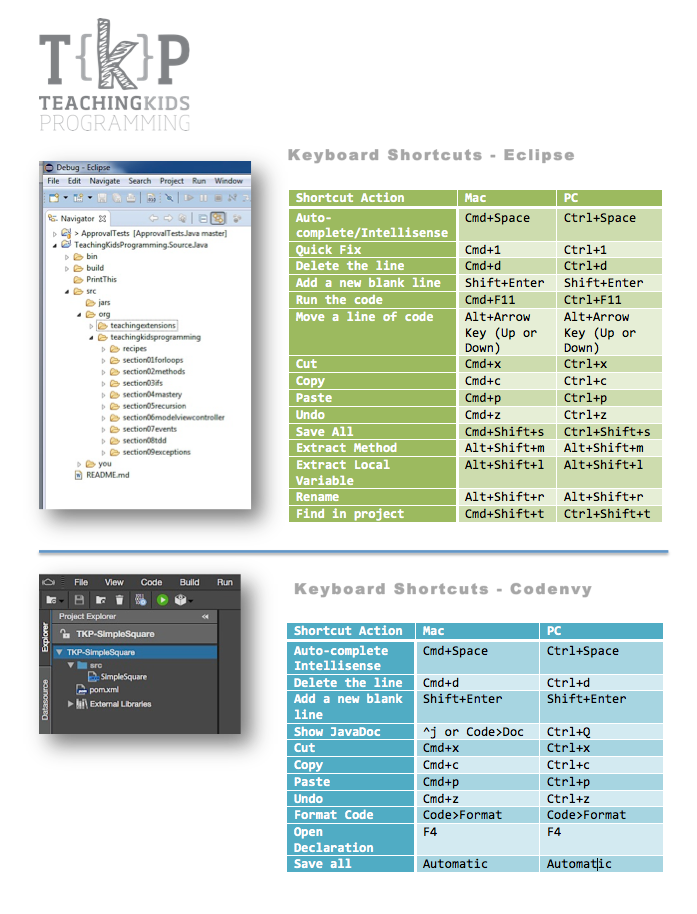
Las recetas son tratadas como una clase de Java con métodos de *stubs ( en el contexto del testeo del software, un trozo de código usado como sustituto de alguna otra funcionalidad. Un stub puede simular el comportamiento de código existente o ser el sustituto temporal para un código aún no desarrollado)*, el cual incluye Español/Inglés en cada línea de código que los niños van a escribir (traducir). Cada archivo de clase en Java también hace referencia a objetos de Java pre-creados, por ejemplo, Tortugas, etc., y se encuentra como un archivo .java en cada lección, por ejemplo, CuadradoSimple.java.

Busca la sección “Programación en Pareja” al final de esta página para una explicación más completa.

 El profesor se prepara (con antelación) para enseñar cada curso al ver el video de TKP en YouTube para la recta que las parejas van a completar. El profesor puede detener el video y preguntar a los estudiantes preguntas sobre el código, con el fin de verificar el entendimiento de los estudiantes. Los videos podrán ser encontrados en el [canal de TKP de YouTube](https://www.youtube.com/user/tkpjava/playlists) y tienen el mismo nombre que las recetas, por ejemplo, ‘Recapitulación de un Cuadrado Simple’. Los videos de resumen son de 10-15 minutos de largo. También hemos creado un documento en PowerPoint que puede ser usado para reforzar las principales construcciones del Lenguaje en Java (revisar la imagen) - que puede ser descargada desde [aquí](https://drive.google.com/a/wandarina.com/file/d/1MnJj1QvxYOELvR4U_L37ydP8-3-nApIE/view?usp=sharing).

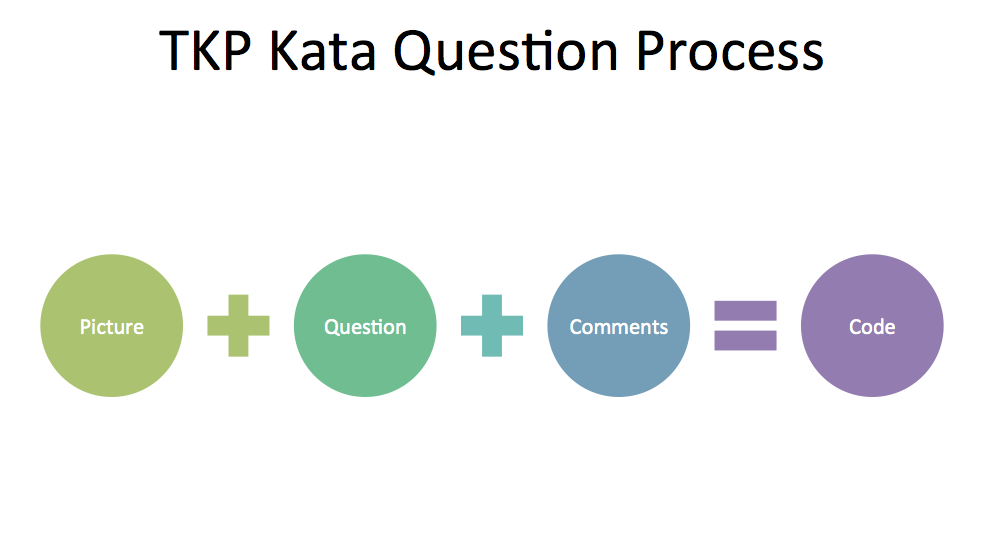
2) **Recapitulación** – En esta sección los alumnos deberán de dejar por un momento sus computadores y prestarán atención a cómo el profesor realizará la receta que ellos recién codificaron. El profesor enseñará esta sección por medio del **método socrático**, (por ejemplo, realizará preguntas, tales como '¿Qué tipeo ahora?' o '¿Cuál es el atajo de teclado para realizar la siguiente acción?', etc...) y codificar toda la receta otra vez, mientras los niños responden las preguntas y observan.

 Tip: El profesor debería de cometer errores a propósito, tal como agregar extras valores a los parámetros, o dejando sin cerrar los corchetes, u omitiendo el símbolo de ‘punto y coma’ (‘;’) al final, y después preguntar a la clase: “¿Cómo corrijo este error?”. También, el profesor debería fortalecer el uso de los atajos de teclado, al preguntar a los alumnos: “¿Cuál es el atajo de teclado para <borrar una línea>, <completar un término>, <realizar un QuickFix>, etc?'. En la siguiente la siguiente imagen se ha incluido una lista de atajos de teclado tanto para Eclipse como para Condenvy, los editores de Java.



**3) Variaciones / Preguntas Kata** – El profesor es el encargado principal de esta lección, los estudiantes trabajan en parejas. El profesor debe de preguntar a los estudiantes el nombre de las propiedades de las recetas completadas o los resultados (para los juegos), por ejemplo, la longitud de la línea, el grosor de la línea, etc. Y se debe de escribir en el pizarrón un tipo de matriz (revisa el ejemplo del cuadro en la receta que se encuentra a continuación). Los profesores deben de verbalmente guiar a las parejas de estudiantes en los cambios de código. Los cambios deben de empezar con simples pasos/cambios, por ejemplo, para cambiar las variables desde un valor de una constante, tal como “azul” a “rojo”, después continúa con cambios más complejos, tales como Colors.Red (Colores.Rojo) a Colors.GetRandomColor() (Colores.AquirirColorRandom()). Instrucciones escritas de como guiar las variaciones pueden ser encontradas en cada plan de curso (lección) para los profesores. Los profesores deben guiar verbalmente a los estudiantes como se describió anteriormente.

 Tip: El profesor puede usar una o más “Pregunta(s) de Kata” para dar comienzo a las variaciones. La idea (véase en la imagen siguiente) es realizar una pregunta sobre el resultado de la receta para lograr que los estudiantes vean (dibujo/draw), formulen (pregunta/question), descomponen (escribir y colocar comentarios en Español/Inglés/write and place English comments) y después traduzcan (codifiquen/code) para crear resultados.



Hemos creado un pequeño video en el cual se presenta también este proceso. Puede ser encontrado [aquí](https://www.youtube.com/watch?v=eSxPP9NZzZ0). Adicionalmente, en cada curso hemos agregado una “Pregunta d Kata” en el principio de cada sección de variación.

Cada curso incluirá un ejemplo de una matriz de variación, un caso se puede observar el la tabla de acontinuación. La matriz de variación tiene por objeto ser un punto de partida para esta sección del curso. Es común y deseable, que los estudiantes experimenten y exploren más allá de las variaciones básicas que el profesor presente.

### **Ejemplo de Matriz de Variación**

|  |  |  |
| --- | --- | --- |
| **Característica** | **Valor** | **Refactor** |
| Lados | 5 | no |
| Longitud | i | no |
| Color | ColorWheel | no |
| Rotación | 1 | n- |
| Ancho | 1-4 | no |
| Fondo | Blanco (por defecto) | exponer por defecto |
| Número de líneas | 200 | no |

Durante la porción de variación de cada curso, el profesor reforzará el refactor del código. Los cursos de TKP comúnmente incluyen el siguiente tipo de refactorización

During the Variation portion of each course, the teacher reinforces code refactoring. TKP courses commonly include the following type of refactoring (usado tanto cuando el maestro enseña las recetas básicas como también durante las variaciones):

### **Tipos de ejemplos de refactorización utilizados en el material de curso de TKP**

|  |  |  |
| --- | --- | --- |
| **Tipo** | **Ejemplo de uso** | **Atajo de teclado** |
| Renombrar | Renombrar objetos | Alt+Shift+R |
| Extraer variable local | Exponer valor por defecto | Alt+Shift+L |
| Extraer variable local | hacer que el valor sea variable y reutilizable | Alt+Shift+L |
| Método de Extracción | extraer parte de la lógica del método existente, generalmente para su posible reutilización | Alt+Shift+M |
| Agregar comentarios | utilizado para agregar marcadores de comentarios | Ctrl + / |
| Variable en línea | (avanzado) utilizado para mejorar la legibilidad del código | Ctrl + 1 |

4) **Examen** – El profesor debe explicar el formato, es decir, traducir una línea, luego ejecutarla para ver si lo hizo bien y luego pedir a cada pareja que trabaje en conjunto para completar correctamente cada línea (o pregunta) del examen. El docente es un **facilitador** en esta sección.

Si la pareja está teniendo problemas en avanzar, el profesor debe de guiar a las parejas para que descubran la respuesta haciendo que vuelvan a leer la línea de Español/Inglés o que utilicen el editor de Eclipse para explorar y leer la documentación (incluyendo el ejemplo de uso de los objetos que se muestran en la documentación, como *ColorWheel.addColor (Colors.Blues.Blue);*).

El profesor debe alentar a los niños a ejecutar su programa después de que hayan traducido **cada** pregunta. El maestro debe recordarles a las parejas que si tienen problemas en avanzar o se encuentran ‘atascados’, pueden usar la opción de 'DESHACER' para regresar a un estado de trabajo. El profesor debe de tratar de evitar decirles a los alumnos la respuesta correcta, prefiriendo guiar a las parejas para que ellos mismos descubran la respuesta adecuada.

El examen para una receta se encuentra en la carpeta CourseNumber del grupo de trabajo de TKP, por ejemplo, el Curso Uno con Receta de Cuadrado Simple, incluye un *archivo de código auxiliar* (stub) de Java llamado Examen de Cuadrado Simple. Los exámenes se llaman con el mismo nombre que la receta, por ejemplo, SimpleSquareQuiz.java y consiste de una clase con apéndices de método, y Español/Inglés para sea traducido por las parejas de estudiantes.

 Nota: Los exámenes de TKP no están diseñados para ser calificados por los maestros. Están diseñados para permitir que las parejas de estudiantes demuestren (y disfruten) el dominio del concepto. Los exámenes están diseñados para que TODOS los alumnos puedan responder TODAS las preguntas correctamente.

**5) Inmersión Profunda:** el profesor debe explicar solo un par de estas preguntas como grupo. En “Inmersión Profunda”, cada método es una pregunta. El docente debería alentar a las parejas a cambiar SÓLO los caracteres subrayados, cuando estén intentando que el método pase su prueba. **IMPORTANTE** El modelo que ejecute cada prueba PRIMERO - debe fallar e indicar a los niños que lean el resultado del error. Luego, cambie SÓLO el guión bajo para que la prueba pase. Ejecute esa prueba para verificar que haya pasado (el resultado será una "barra verde" en el ejecutor de prueba JUnit).

El docente debe usar los mismos pasos para el cuestionario, con el fin de facilitar el guiar a los niños a través de los ejercicios de Inmersión Profunda. Los estudiantes pueden trabajar en parejas o en un formato **Randori** (Randori también se llama 'Programación de la mafia' o “Mob Programming”). Se sugiere hacer rotar los pares luego de que cada método se cumpla. Si se usa un formato Randori, los estudiantes deben rotar después de cada ejecución de prueba exitosa. En cualquier caso, el maestro debe alentar a compartir a los alumnos, al menos una cosa que aprendieron en la prueba de obtención del método para aprobar.

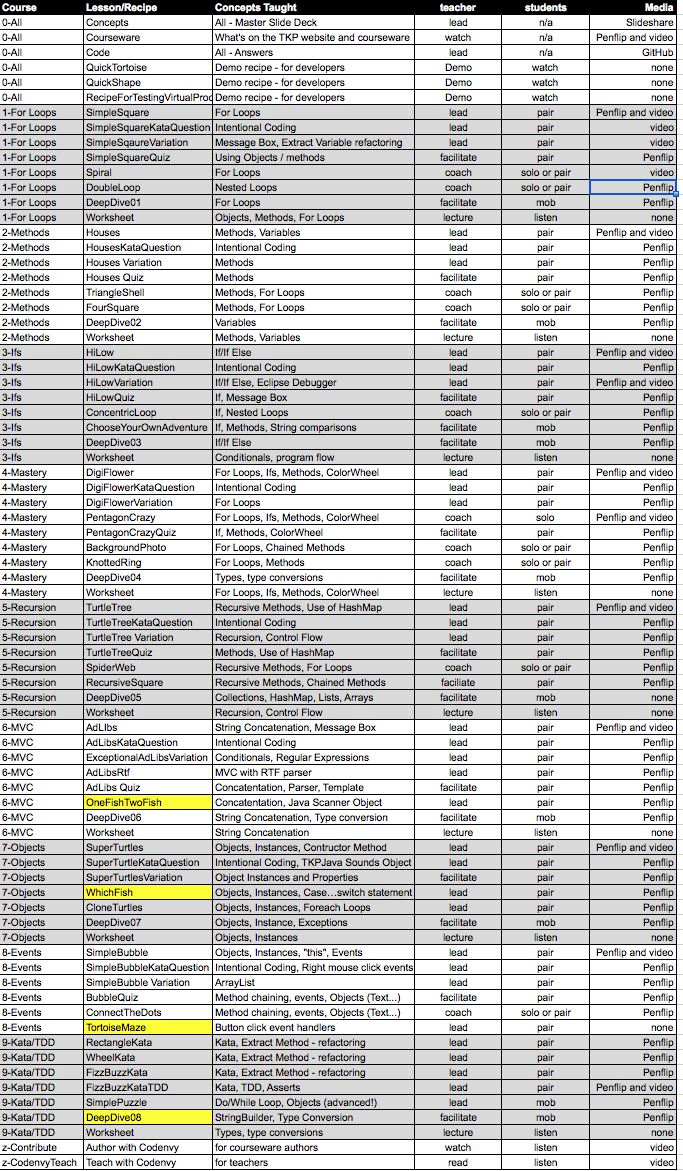
 Tip: En el siguiente enlace, se puede encontrar más información sobre 'Mob Programming' - <http://www.agileconnection.com/article/mob-programming-whole-team-approach>

La información sobre Inmersión Profunda se encuentran en la carpeta Número de Curso del espacio de trabajo de TKP para el curso actual, por ejemplo, CourseOne, CourseTwo. Están enumerados para coincidir con el curso TKP, es decir, DeepDive01.java, etc ... están diseñados para ejecutarse con el corrector de prueba en Eclipse e incluyen instrucciones en inglés al principio del archivo de Java sobre cómo usar los atajos de teclado para ejecutar el archivo usando el corredor de prueba JUnit.

**6) Hoja de trabajo:** los cursos de TKP incluyen hojas de trabajo que pueden ser impresas. El objetivo de estos folletos es permitir que los maestros refuercen los conceptos principales de programación (en particular la terminología, como por ejemplo: clases, métodos, variables, etc.) que se han enseñado en cada curso. Una forma de aplicarlos es, solicitando a los estudiantes que identifiquen las ocurrencias de determinados tipos de objetos encontrados en las soluciones de recetas y, que registren sus respuestas en una hoja de trabajo. Puede descargar las hojas de trabajo desde esta [ubicación](https://github.com/swandarina/TKP-WANDARINA-Recursos_en_Java/tree/master/src/main/resources/documents).

**7) Xtras**: los cursos de TKP incluyen 'mini recetas' adicionales, como el ESPIRAL, que un maestro puede usar para enseñar el CUADRADO. Estos son complementos que pueden durar entre 10 a 15 minutos. Los profesores pueden dar estos complementos a los equipos que ya hayan terminado el trabajo guiado (exámenes, Inmersión Profunda), antes de pedirles que hagan otra cosa más. Los maestros pueden hacer que las parejas trabajen solos en estos "complementos" durante la clase o pueden pedirles a los alumnos que de forma individual completen este trabajo durante el tiempo de clase libre o en casa.

En la siguiente imagen hemos copiado una lista de todos los cursos y materiales. La lista también coincide con los conceptos computacionales que se enseñan a las actividades de TKPJava. Y enumera el método preferido para enseñar cada actividad (es decir, dirigido, guiado, estudiantes de dos en dos, estudiantes en con el Método Mob, etc.). Los artículos en amarillo están en desarrollo, activo a partir de enero de 2016.



A continuación se presenta un video de una charla en el que se explica el diseño del Material Didáctico de el Java de TKP, expuestos por Lynn Langit, en Oredev, Malmo, Suecia, en Noviembre del 2014.

[](https://www.youtube.com/watch?v=bD32gGTSYQs)

## **Cómo utiliza TKP las prácticas de programación profesional**

## El diseño instructivo de TKP se basa en muchas de las mejores prácticas y principios de Agile (y XP o Extreme Programming). Estos incluyen el uso de prácticas básicas de Agile en la enseñanza (o entrega) de cursos de TKP. El material didáctico de TKP incluye ejemplos de cómo usar estas prácticas en ciertas situaciones durante las clases, y sugerencias sobre cómo incorporar estas prácticas a la enseñanza de los niños al programa. Incluyen lo siguiente:

* **Programación en parejas:** tanto los estudiantes como los maestros trabajan en pares para aprender / enseñar el material de TKP si es posible. El material didáctico de TKP puede ser enseñado por un solo maestro, sin embargo, recomendamos la enseñanza en equipo si es posible. Para los estudiantes, se requiere que estén organizados, cada 2 estudiantes por 1 computadora: una que escribe, otra que indica qué hacer, y en rotación constantemente (de manera general, cada 5 minutos), o la tarea (por ejemplo, cuando una línea o sección de código se completa con éxito).
* **No Gran Diseño Inicial (No Big Upfront Design)**: se aconseja al docente cómo guiar a las parejas de los alumnos a escribir y ejecutar su primer programa dentro de los 5 minutos posteriores al inicio de cada clase de TKP.
* **Desarrollo basado en pruebas:** el material didáctico está escrito para que los estudiantes puedan ser guiados para traducir una línea de Español/Inglés en una línea de código y luego ejecutar (o ejecutar) el resultado, para que las parejas puedan observar si han completado la traducción correctamente. Este es un tipo de desarrollo guiado por pruebas visuales. Además, el método de enseñanza TKP recomienda eliminar los comentarios originales en Español/Inglés, DESPUÉS de que la línea haya sido traducida correctamente.
* **Paso sostenible:** el profesor debe prestar atención especial al ritmo de la clase. Las parejas de estudiantes deben de rotar con el computador, ya sea al completar la tarea o en un intérvalo de tiempo regular (como por ejemplo, 5 minutos). Las parejas también se cambian al final de cada lección (si la clase de ese día es una clase de varias lecciones), para facilitar aún más la transferencia de conocimientos. El material didáctico TKP está diseñado para ser enseñado a través del aprendizaje basado en el dominio (mastery-based learning), es decir, que cada alumno domine cada concepto en cada curso, antes de que empiece el próximo curso.
* **Retroalimentación inmediata:** además de la retroalimentación visual inmediata que obtienen los niños después de ejecutar cada línea de código en la receta, el grupo TKP recomienda la compañía de supervisores (ayudantes) en el aula, para apoyar que las parejas conserven el ritmo. Además de los supervisores en las aulas, el material didáctico TKP incluye un Virtual Proctor, que proporciona retroalimentación visual de todas las parejas de los alumnos al profesor.
* **Artesanía**: cada receta o lección contiene varias secciones, de modo que los estudiantes puedan dominar los conceptos que se enseñan, los APIs y las herramientas, antes de conocer nuevos conceptos, otras herramientas y APIs. Esto también se denomina aprendizaje "basado en el dominio" ('mastery-based') como se mencionó anteriormente.
* **Randori:** un método para enseñar a un grupo con una computadora y un proyector. El grupo se sienta en un círculo y gira cada vez que una tarea se completa con éxito (codificada). Una persona está escribiendo en el teclado (controlador) y la otra persona le está diciendo al conductor qué escribir (navegador). El grupo puede ayudar al navegador si esa persona no está segura o "se queda atascada".
* **Code Katas:** un método de enseñanza al proporcionar abstracciones de mayor nivel para definir el problema que se codificará en una receta típica de TKPJava. Por ejemplo:

|  |
| --- |
| // Here's the code for a kata // Make a square   // Here's line -by-line instructions as in TKP recipes/lessons // Show the tortoise // Do this 4 times // Move the tortoise move 50 pixels // Turn the tortoise turn right 90 degrees |

* **Los códigos Katas** se parecen más a los requisitos del software del mundo real o a las historias de los usuarios, que las instrucciones de línea por línea en Español/Inglés. El proceso de traducir o desglosar conceptos de nivel superior en varias líneas de comentarios en Español/Inglés PRIMERO y luego escribir el código del programa para ejecutar cada una de esas instrucciones en Español/Inglés, una línea a la vez (y probar la corrección ejecutando el programa o una prueba) es clave en el Método Intencional de TKP para enseñar a programar.
* \*\* Código Koans \*\*: un método de enseñanza que utiliza codificación y retrospección (discusión que se da DESPUÉS de codificar con éxito cada método) para enseñar conceptos de programación, TKP usa koans (llamados "inmersiones profundas Profundiza más") para explorar los conceptos de programación con mayor profundidad. Esta sección fue inspirada por los [Ruby Koans](http://rubykoans.com/).

Los TKP de Koans se basan en el desarrollo impulsado por prueba (TDD, test-driven development) y pruebas unitarias. Una prueba de unidad es un método escrito para comparar los valores de salida esperados y los reales de otro método. Las pruebas unitarias a menudo se escriben con el método 'afirmar'. Otra forma de entender el método de Koans es como una forma de 'aprender experimentando: literalmente, se anima a las parejas a' hackear '(o adivinar) para tratar de obtener la respuesta correcta (o pasar la prueba de unidad [o producir una' barra verde '] ]). Un componente clave del trabajo con Koans de TKP es que el facilitador / maestro debe pedir a cada par de estudiantes que reflexionen sobre lo que han aprendido CADA vez que obtienen una prueba de la unidad (prueba unitaria) para que se apruebe un método.

Por ejemplo, en la receta principal de CLASSES TKP de Java, se presenta el concepto central del ciclo 'for'. En las CLASES / inmersión profunda profundiza más, los ejercicios incluyen todos los componentes opcionales de un bucle 'for', tanto como el valor de una sola opción (such as step value) un paso.

A medida que trabajan a través de los koans, los estudiantes observarán una variedad de cosas sobre el lenguaje, el entorno, el arnés de prueba, etc. Los profesores pueden querer escribir las observaciones de los alumnos en una pizarra, rotafolio, etc. durante esta parte de cada curso de TKP.

## **Por qué programar en parejas**

## La programación de pares es una práctica de desarrollo profesional ampliamente utilizada. Muchos estudios han demostrado que esta técnica da como resultado un software de mayor calidad (menos errores) Y desarrolladores más felices.

**Técnica:**

La técnica de trabajo en parejas toma la forma de un estudiante en el teclado y el otro estudiante sentado a su lado. El estudiante en el teclado (conductor) está tomando instrucciones del estudiante a su lado (navegador). Un temporizador simple (a menudo usamos alguna aplicación de un teléfono móvil) se usa para controlar las rotaciones, por lo general cada 5 minutos, pero puede variar, a veces "el rotar al completar la tarea" puede ser lo mejor. Si hay dos profesores, ellos también deberían rotar para ser parte del modelo programación en parejas. Ejemplos de métodos alternativos para rotar las parejas de estudiantes, se da cuando rotan después de lograr cumplir cada tarea (como durante el Examen de TKP o Inmersión Profunda de TKP).

**Métodos:**

**Preparar:**

El trabajo en pareja permite el doble de estudiantes por computadora. Reduce la cantidad de computadoras que el docente debe configurar, aunque es muy importante tener en cuenta que el aula necesitará la misma cantidad de sillas en comparación con la cantidad de estudiantes.

**Unidad de clase:**

La unidad del grupo es importante porque permite que la clase esté en la misma página. Es más difícil tener unidad cuando los estudiantes trabajan independientemente, especialmente cuando se trata de un grupo grande de estudiantes. El trabajo en parejas reduce efectivamente la cantidad de estudiantes que trabajan independientemente en ½.

#### **Ecualización de habilidades:**

#### Los estudiantes ingresan con diferentes niveles de habilidades, lo que dificulta que un maestro les enseñe simultáneamente. El trabajo en grupo ayuda a nivelar el ‘campo de juego’ para todos los estudiantes. La mejor parte es que no penaliza a los estudiantes más avanzados porque a medida que explican el material a sus compañeros de clase, obtienen una comprensión más profunda.

#### **Enfoque:**

#### Trabajar con un compañero ayuda a ambos estudiantes a concentrar sus energías en la codificación y reduce las distracciones (por ejemplo, en la navegación web, etc.).

**Social:**

El trabajo en parejas ofrece una atmósfera más amigable y sociable, y con esto pretendemos contradecir la idea errónea de que la programación está asociada con ser antisocial. Esto es especialmente importante para las niñas, que generalmente trabajan mejor en grupos donde pueden comunicarse y debatir entre sí.

**Reduce frustraciones:**

En la programación, encontrar errores causa una enorme frustración para todos. Cuando los estudiantes trabajan individualmente, es fácil pasar por alto bugs o errores. Sin embargo, con dos pares de ojos trabajando juntos atentamente, la cantidad de bugs o errores presentes se reduce enormemente. Muchas veces un concepto puede ser confuso para uno de los estudiantes de la pareja, pero no para el otro. Al trabajar juntos, sus frustraciones se reducen.

**Química Corporal:**

Las rotaciones son implementadas por el docente porque después de 15 minutos de estar sentados, la química corporal cambia y se ralentiza. Al exigir que los niños se pongan de pie cada 15 minutos, ayudamos a evitar que se produzca este cambio en su cuerpo. Esta es la razón detrás de por qué el docente debe abstenerse de dejar que los alumnos permanezcan en su posición original cuando llegue el momento de la rotación o el cambio de parejas.

**Mayor visión periférica:**

Según los estudios de ciencias cognitivas, las emociones negativas dificultan la capacidad de participar en una percepción visual más amplia. (http://www.unc.edu/peplab/broadening.html)

Por ejemplo, un estudio comparó el tiempo que las personas pasaron mirando una imagen de acuerdo con su estado emocional. Cuanto más infeliz se siente, es menos probable que el individuo pueda ver toda la pantalla. Es difícil aprender cuando uno no puede ver todas las opciones disponibles para ellos.

**Creatividad:**

Los estudios en ciencia cognitiva también han presentado el hecho de que las habilidades creativas de uno pueden ser limitadas cuando se trabaja en soledad. (http://ascc.artsci.wustl.edu/~ksawyer/groupgenius/). Sin embargo, la creatividad puede florecer cuando a uno se le da la oportunidad de intercambiar ideas con los demás.

**Durante la clase**

Los niños trabajan en parejas a menos que se indique lo contrario. En intervalos de 5 minutos (use un temporizador), las parejas se preparan para intercambiarse. Un está en el teclado, mientras que el otro le dice qué escribir.

En la imagen inferior se otorga un ejemplo niños organizados en parejas, mientras aprenden el Java de TKP.



## **Por qué programación en grupo (Mob)**

## Otro método de enseñanza que utilizamos es la programación en grupo (o mafia). Este es un proceso con una computadora, que está conectada a un proyector. Un estudiante se sienta en el teclado ("el conductor") y escribe lo que el otro alumno que está de pie ("el navegador") le dice a ese alumno que escriba. El resto de los estudiantes se sientan en un círculo de sillas alrededor del conductor y el navegador.

Para cada problema de codificación, primero el navegador intenta resolverlo. El conductor teclea lo que se le dice que escriba y luego ejecuta el programa para verificar que el resultado sea correcto. A continuación, el docente (que actúa como facilitador durante este proceso) solicita a AMBOS, al conductor y al navegador, que compartan al menos un aspecto de la codificación, de las herramientas, del lenguaje, etc. que ambos aprendieron al resolver el problema. El facilitador puede registrar opcionalmente la información del alumno en una pizarra o rotafolio.

Después de resolver un problema, todos los estudiantes se ponen de pie y rotan en el círculo, para que cada alumno tenga al menos un turno como conductor o navegador durante el ejercicio.

Para obtener más información sobre los beneficios de la programación de mafia, consulte esta referencia:

<http://en.wikipedia.org/wiki/Mob_programming>

## **TKP Classroom Management Tips y la utilidad TKP Virtual Proctor**

## Incluimos esta sección para los profesores de nuestros cursos que son nuevos en liderar un aula llena de niños. Los maestros de K-12 con experiencia pueden desear omitir esta sección ya que probablemente estén familiarizados con la mayoría de estas técnicas.

#### **Pequeños pasos**

* Traduzca la línea siguiente en la que se pueda ver el resultado de la ejecución- el profesor guiará a los estudiantes a través de las recetas en el orden en que se puedan VER los resultados, NO en orden ordinario (1,2,3, etc). Todas nuestras recetas sugieren el ordenar las líneas para que sean enseñadas. (All of our recipes have suggested line orderings on the English lines for teaching, i.e. teach line 2 first, teach line 5 second, etc…
* Si no se puede ver una sola línea, use el método 'Fake it until you make it' (Finge hasta que lo consigas), para continuar con las otras traducciones, como por ejemplo, coloque el círculo donde está el mouse (no necesita un círculo, que necesite un radio, o un mouse) *ShapeMaker. CenterShapeAt (ShapeMaker.CreateCircle (20), 20,20)* ". Esta acción pone un círculo en la pantalla, ahora vamos a trabajar en que sea el círculo correcto, y usando las ubicaciones del mouse ..."
* i.e. Place the Circle where the mouse is (you don’t have a circle, which needs a radius, or a mouse) ShapeMaker.CenterShapeAt(ShapeMaker.CreateCircle(20), 20,20) “ok, this puts a circle on the screen, now lets work on it being the correct circle, and using the mouse locations…”

**Tipeo del instructor**

* Siempre tipee lo que sea suficiente, para mostrar la información que será de mayor utilidad para los niños.
* Si los estudiantes están buscando por alguna acción que la tortuga debería realizar, tipee para que de esta forma, la lista de métodos aparezca en el lateral.
* De a los niños el tiempo suficiente para explorar la documentación
* Siempre espere hasta que la mayoría de las parejas haya conseguido un código que esté funcionando (puede revisar los resultado de proctor), antes de que tipee el código correcto y lo ejecute.
* Está bien que el docente permanezca callado y permita a los niños trabajar. No, realmente.
* Es una buena idea, ocasionalmente, cometer errores cuando el docente esté escribiendo el código, ya sea escribiendo exactamente lo que dice el ejemplo / documentación dicen, y luego corrigiéndolo, porque se puede enseñar que lo que necesitamos es ligeramente diferente (SetPenColor ("rojo") / Para i = 1 a 10), o para obtener algo que no se compila (por ejemplo, durante la recapitulación, intente escribir Tortoise.Turn (izquierda) -Tortoise.Turn(left)-y luego, cuando no se compile, señale el número de línea y el mensaje de error, además de la documentación en el lado otra vez). and then when it doesn’t compile, point out the line number and the helpful error message, and then to the documentation on the side again).

**Guía — No lo digas**

* Pídales a los niños que lean en voz alta las líneas en Español/Inglés: use esta técnica con frecuencia.
* Si los niños están estancados, pídales que vuelvan a leer la línea en Español/Inglés correspondiente.
* Exploración, haga que los niños usen las flechas para descubrir qué hacer. Si no saben por dónde comenzar, **Ctrl + Space** mostrará todas las posibilidades.
* Si alguien pregunta "¿es esto correcto?", Dígales "ejecútalo".
* Si quieren saber cuál método usar, señálesle la documentación.
* Refrene el instinto de decirles la respuesta correcta, pero ayúdelos para que sigan avanzando.
* Observe que los dos miembros de la pareja estén contribuyendo, si no, solicite al miembro que no está participando, que se involucre.

#### **Crea y usa puntos para guardar**

#### Al menos, al final de cada receta, todos deben guardar su trabajo

#### Para volver al comienzo de una lección rápidamente, vaya al Navegador en Eclipse, haga clic derecho en el archivo Java y luego haga clic en 'Revertir usando historial local' (Revert using local history’ ) y luego haga clic en la primera entrada, esto volverá a establecer la lección en el punto de inicio

#### 

#### **Recapitulación**

* Señale los matices del editor, la documentación, la lógica, los accesos directos sobre la marcha.
* Use este tiempo para Explicar los Conceptos que acaban de Experimentar.
* Introduzca la Programación en Lingo, por ejemplo, para el concepto de CUADRADO o SQUARE, puede decir 'los sustantivos son objetos y se muestran con cubos; los verbos son operaciones y se muestran con ruedas, etc.
* Este es un buen momento para hacer un diagrama de alcance (por ejemplo, enumerar objetos, métodos y variables que se han utilizado durante la lección)
* Este es un buen momento para cometer errores a propósito, como "Tortoise.SetPenColor (azul)" y pedirle a los niños que le digan cómo corregir los mismos

#### 

#### **Tips para Codificar (Depurar)**

#### DESHACER: Aconseje el uso de deshacer primero

#### FORMATO: hemos configurado Eclipse para formatear automáticamente al guardar.

#### MÉTODOS (Operaciones): verifique los pares de paréntesis ‘()’ y los argumentos -haga que los niños hagan clic en el método (palabra) en el editor, y que luego lean la documentación-, particularmente en el ejemplo.

#### BLOQUES: verifique por la ubicación de las llaves ‘ {}’ y / o falta de ellas

#### CUERDAS: verifique por las comillas dobles ‘ “ ” ’

#### LAZOS INTERMINABLES: regrese de nuevo a un estado ejecutable, esto es generalmente el resultado de poner una cadena de texto en los argumentos para un lazo

#### DEPURADOR: use el Depurador de Eclipse, a continuación un breve video en YouTube sobre cómo hacerlo: <https://www.youtube.com/watch?v=M8F6ziuccmw>

#### 

#### **Trabajo en Grupo y Energía**

#### 

#### Escoja las parejas de “niños lentos” con niños rápidos; asegúrese de rotar las parejas que progresan con lentitud. No juntes hermanos. Si los adultos quieren tomar la clase, haga parejas de adultos.

#### Conductores del asiento trasero: si un niño que debería estar navegando está tipeando, interrúmpalo y recuérdele al niño que NO puede tocar el teclado cuando está con una función diferente.

#### Rote a los profesores, así como a los niños

#### Rote cada 5 minutos o al finalizar una tarea

#### Haga que se pongan de pie al rotar de compañeros, este movimiento simple, ayuda a darles más energía y atención.

#### Los niños deben de ya estar escribiendo dentro de 3 primeros minutos del inicio de la clase. Es muy importante empezar cada clase con alguna acción rápidamente.

#### Hágale a los niños MUCHAS preguntas, todo el tiempo. No les anticipe, pregúnteles: es fundamental para mantenerlos comprometidos.

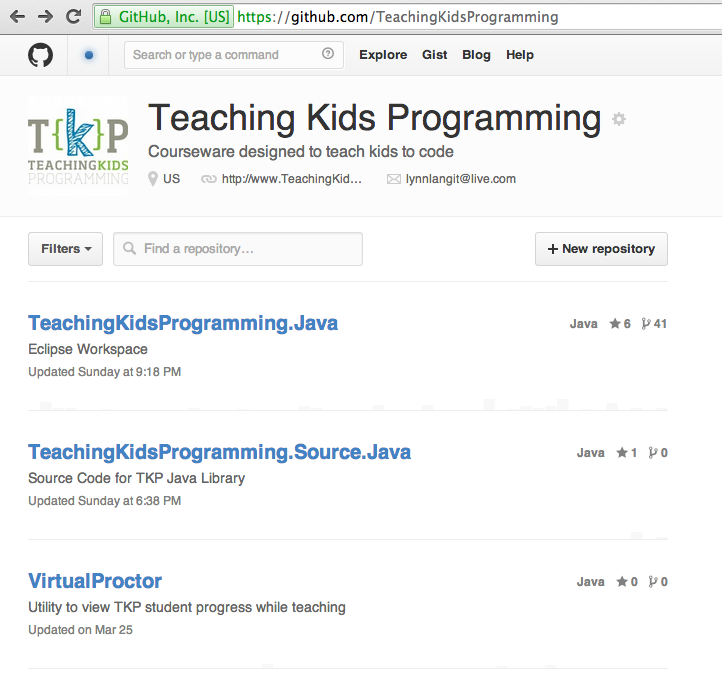
#### Aliente un ‘caos controlado’: el ruido es bueno, los estudiantes deben estar hablando / riéndose;

#### Durante las variaciones, los niños deben estar alegres.

#### Si enseña después del almuerzo, dedique un tiempo extra a aumentar la energía antes de sentarse.

#### Cambiar de asientos debería de ser una actividad de alta energía. NO levante las manos ANTES de hacer esto, ya que disminuye la energía en la habitación.

## **Guía del material didáctico de el Java de TKP y el código fuente en Github**



El material del curso es de código abierto. La versión para enseñar (que debe de ser instalada en las computadoras de los estudiantes, consta de un archivo .jar de Java y algunos archivos de servicio. Los materiales didácticos del curso de TKP se encuentra en el repositorio de Github llamado

 <https://github.com/swandarina/TKP.Java>

#### 

#### **Acerca de la herramienta TKP Virtual Proctor**

#### Además del material didáctico, TKP tiene dos repositorios relacionados en Github. El primero se llama [TeachingKidsProgramming/VirtualProctor](https://github.com/TeachingKidsProgramming/VirtualProctor). Esta es una aplicación simple que utilizamos para ayudarnos a controlar el progreso de los estudiantes en el aula. Para usar esta utilidad, solo abra un navegador para <http://virtualproctor.tkpjava.org>

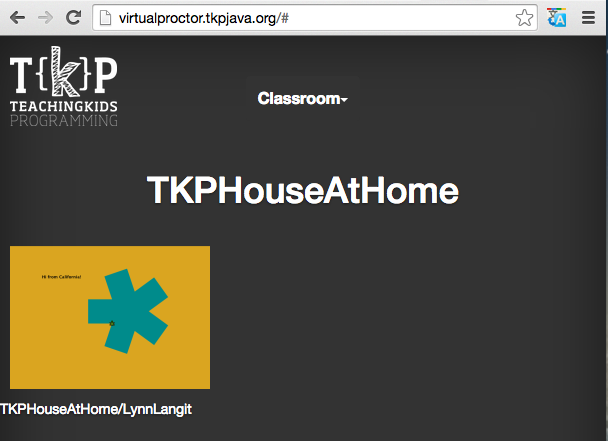
#### 

#### Por defecto, el TKP VirtualProctor toma una captura de pantalla de las ventanas del programa (gráficos) DESPUÉS de que el alumno cierre la ventana y envíe esa captura de pantalla a un servicio basado en la nube. El sitio web asociado, muestra la última captura de pantalla con el nombre predeterminado de la computadora.

#### Para configurar VirtualProctor con un nombre personalizado de la máquina de un alumno, agregue la siguiente línea de código a cualquier receta de TKP:

|  |
| --- |
| VirtualProctor.setName("studentsName");  VirtualProctor.setClass("classroomName"); |

Después de agregar esta línea de código, ejecute el mismo y cierre la ventana del programa (gráficos). TKP Virtual Proctor contiene un intervalo de actualización automático. Puede arrastrar y soltar las pantallas en la página del navegador para que las pantallas coincidan con el diseño físico del aula. Una ejemplo de esta pantalla se muestra a continuación.



El código fuente (versión en Español/Inglés) de los APIs que escribimos para usar en el material didáctico de TKP se encuentra en el repositorio de Github en https://github.com/swandarina/TKP.WANDARINA.Fuente.Java.

Aquí se puede encontrar qué API se escribió (o amplió) para crear el material didáctico. También puede ser de interés para los maestros en este repositorio [la sección respuestas completas](https://github.com/TeachingKidsProgramming/TeachingKidsProgramming.Source.Java/tree/master/src/org/teachingkidsprogramming/recipes/completed) de la receta. Deliberadamente ocultamos la ubicación de las respuestas, ya que no queremos que los alumnos copien y peguen las respuestas, preferimos que exploren el API y, con la guía del docente, codifiquen los resultados ellos mismos.

## **Guía para escribir recetas de escritura TKP**

## Nos esforzamos mucho para crear recetas. Probamos e iteramos constantemente, para que los niños tengan una gran experiencia aprendiendo. Algunos de los conceptos básicos que tratamos de seguir al crear recetas son los siguientes:

1) Las recetas deben ser DIVERTIDAS y COMPROMETEDORAS

2) Las recetas deben poder completarse entre 15 y 20 minutos

3) Las recetas deben tener entre 20 y 40 líneas de Español/Inglés (y no más)

4) Las recetas deben enseñar solo de 1 a 3 conceptos de programación básica

5) Creamos tanto la receta como la variación (a menudo también el examen) cuando creamos una nueva receta

6) El proceso es el siguiente:

* Hacerse una idea, por ejemplo dibujar un objeto, escribir algo en la pantalla, etc.
* Codifíquelo - ver qué objetos, propiedades, métodos o eventos ya existen en el idioma
* Cree algunos objetos nuevos (u objetos envueltos) en la biblioteca el Java de TKP (en Github), como 'Tortoise.SetPenColor' para encapsular 'ProgramWindow.PenColor' para que los objetos de la lección sean más visibles para los niños
* Escriba la documentación, incluyendo ejemplos de código para sus nuevos objetos en los objetos, métodos, propiedades, etc., de el Java de TKP
* Cree el 'código auxiliar' para la receta, que es el método principal y cualquier otro método que quieras que los niños completen.
* Agregue los comentarios en Español/Inglés a la receta de 'código auxiliar'
* Determine el 'orden de ejecución' y agregue los números de línea al final de cada línea de Español/Inglés
* Pruebe la receta con niños reales
* Realice cambios y actualizaciones según los comentarios de su probador (niños)
* Edite el código fuente de TKP de Java en Github (obtendrá su propia rama para actualizar)
* Agregue su nueva receta a su institución de TKP y cree una solicitud de extracción para que podamos ver tu receta
* Cree materiales de capacitación para docentes (hojas de trucos [claves de respuesta] y videos)
* Agregue información al número de curso de TKP en Google Docs sobre cómo enseñar la receta

**Pautas de redacción de recetas (para los comentarios en Español/Inglés de las recetas de Java)**

Gracias a la colaboradora Katrina Owen por esta guía

*'Notas de formato de receta*

*'Agregue líneas vacías después de cada instrucción*

*'Agregue dos líneas vacías antes de comenzar una receta*

*'Agregue espacios después del apóstrofe para endentar comentarios*

*'Invoque el método de cuatro espacios*

*'Invoque los lazos de cuatro espacios*

*'Siempre termine los bucles con 'Repetir*

*'Empiece una receta de método de esta forma:*

‘————- Receta para AlgunNombreDeReceta

'Finalice una receta de método como esta:

'----- Fin de la receta de AlgunNombreDeReceta

'Use Español/Inglés, no código, por ejemplo, en lugar de 'Set/Establecer el PenColor en' ReallyBrightRed '

'Use 'Cambie el color de la línea que dibuja la tortuga a rojo muy brillante

'corolario:

'No ponga mayúsculas en los nombres de los objetos en los comentarios

'No ponga citas amenazantes sobre nombres de colores

'Nombres de colores con minúsculas (y no los encadenen juntos)

'Comiencen nombres de variables con ' la actual x es y '

'Siempre formatee el programa al final para deshacerse de los espacios en blanco en líneas vacías.

### **Influencias / influenciadores del curso didáctico de TKP**

### Se nos ha pedido que creemos una lista de influencias / influenciadores en el material didáctico TKP. Aquí hay un enlace a un Gist con esa información:

<https://gist.github.com/lynnlangit/15e12d902ba66654468b>

## **El material didáctico de TKP para otros lenguajes de programación: SmallBasic, C #, T-SQL y más**

## Nuestro objetivo principal es escribir material didáctico para niños de secundaria en Java. Sin embargo, hemos experimentado con otros lenguajes de computadora a lo largo de los años. Utilizamos nuestro 'Método intencional de TKP' de enseñanza en todos los dominios. Para ello, enumeraremos enlaces a nuestros otros repositorios de cursos a continuación:

## 1) SmallBasic en [Codeplex](https://extendsmallbasic.codeplex.com/) y en [YouTube](https://www.youtube.com/watch?v=7ivcutYEF_I&list=PLC8CE7C642AB41E35)

## 2) C # en [Pluralsight](http://www.pluralsight.com/training/Courses/TableOfContents/teaching-kids-programming)

## 3) Kodu en [Slideshare](http://www.slideshare.net/lynnlangit/kodu-class) y en Lynda.com (<http://www.lynda.com/search?q=kodu>)

## 4) T-SQL en [Codeplex](https://datinggame.codeplex.com/)

**Acerca de TKPJava y el examen APJava CS**

A través de los comentarios de los maestros, en TKP escuchamos que nuestro material didáctico el Java de TKP se está utilizando como parte de la preparación para los estudiantes antes de que tomen el examen APJava CS.

Si bien no diseñamos nuestro material didáctico para que responda específicamente con los objetivos del examen, proporcionaremos un cuadro para enumerar las áreas de concentración del examen y nuestros objetivos del curso.

[Aquí](https://secure-media.collegeboard.org/digitalServices/pdf/ap/ap-course-overviews/ap-computer-science-a-course-overview.pdf) hay una lista más completa de los objetivos del examen AP.

**Objetivos del examen: objetivos de AP Computer Science A**

Los estudiantes deberían poder:

• Diseñar, implementar y analizar soluciones a problemas;

• Usar e implementar algoritmos comúnmente empleados;

• Desarrollar y seleccionar algoritmos apropiados y estructuras de datos

para resolver nuevos problemas;

• Escribir soluciones con fluidez en un paradigma orientado a objetos;

• Escribir, ejecutar, probar y depurar soluciones en la programación del lenguaje de Java, utilizando el estándar de clases e interfaces de la biblioteca de Java

del subconjunto AP Java;

• Leer y comprender programas que constan de varias clases y objetos que interactúan;

• Leer y comprender una descripción del proceso de diseño y desarrollo que conduce a dicho programa; y

• Comprender las implicaciones éticas y sociales del uso de la computadora.

**Esquema del tema para AP Computer Science A**

**Diseño de Programa Orientado a Objetos - cierta cobertura en TKPJava**

A. Programa y diseño de clase - Curso 6-8

**Implementación del programa: cierta cobertura en TKPJava**

A. Técnicas de implementación: todos los cursos

B. Construcciones de programación - todos los cursos

C. Clases de biblioteca de Java e interfaces incluidas en el subconjunto de Java de AP: parcial

**Análisis del programa: cierta cobertura en TKPJava**

A. Pruebas - Curso DeepDives y Curso 8

B. Depuración - opcional, se puede incluir como dirigido por el profesor

C. Excepciones de tiempo de ejecución - Curso 6

D. Corrección del programa: todos los cursos

E. Algoritmo de análisis

F. Representaciones numéricas de enteros - DeepDives and Courses 6-8

**Estructuras de datos estándar: cubiertas en TKPJava**

A. Tipos de datos primitivos (int, boolean, double) - Cursos 1-8

B. Cadenas de texto (string) - Cursos 1-8

C. Clases - Curso 7-8

D. Listas - Cursos 5 y hasta

E. Matrices (1-dimensional y 2-dimensional) - Cursos 7 en adelante

**Operaciones estándar y algoritmos: cierta cobertura en TKPJava**

A. Operaciones en estructuras de datos - Curso 7 TKPJava

B. Buscando

C. Clasificación

**La computación en contexto - no está cubierta en TKPJava**

A. Confiabilidad del sistema

B. Privacidad

C. Asuntos legales y propiedad intelectual

D. ramificaciones sociales y éticas del uso de la computadora