Proyecto: Creación de Examen.



Asignatura: programación.

Profesor: Gonzalo Honores.

Alumnos: Matias Barra, Edinson Godoy y Conzuelo Ardiles.

Fecha. 25-10-2023.

Dentro del presente texto se explicará el proyecto el cual consistía en la creación de un examen usando el lenguaje de programación, Java, Se podrá encontrar la especificación de la creación, el análisis y los pasos que seguimos para poder completar este proyecto.

Para comenzar debo explicar las indicaciones de este proyecto debía ser una actividad por consola la cual, junto con la aplicación de conceptos de la programación orientada a objetos, pudiera construir distintas clases en Java para la creación de un examen el cual consistía en un tipo cuestionario con 3 tipos de preguntas, verdadero y falso, selección múltiple y respuestas cortas. Estas preguntas contaban con un puntaje el cual representa el resultado del examen, las preguntas no deben ser mas de 10 y debe contener unas 3 preguntas por cada ítem. Finalmente se debe administrar el porcentaje de respuestas y como extra tener un registro de los porcentajes obtenidos.

Para finalizar este proyecto contamos con unas 3 semanas de plazo, las cuales se van avanzando semana a semana usando de repositorio GitHub para mantener la conexión con todos los compañeros.

Luego de tener las indicaciones del proyecto comenzamos con el modelado de la solución, comenzamos definiendo las clases que dividirían las funciones para la creación del examen, las cuales a grandes rasgos son descritos:

* ExamDemo.java el cual será el main y donde se inicializaría el examen.
* Exam,java el cual agrega preguntas y calcula el porcentaje obtenido.
* Pregunta.java la cual creara una estructura básica para representar preguntas con un texto y un puntaje asociado junto con un método buscar para verificar las respuestas del usuario. Esto con el objetivo de crear clases hijas de las preguntas especificas las cuales obtengan la herencia de pregunta.java
* TFpregunta.java será una clase hija de pregunta.java que permite a los usuarios responder preguntas verdadero/falso y verifica si sus respuestas son correctas o incorrectas.
* Resp\_Cortas\_Pregunta.java obtendrá la herencia de pregunta y será diseñada para representar respuestas cortas donde se proporcione una respuesta de texto.
* Selec\_mul\_Pregunta.java clase hija de pregunta.java la cual representa preguntas de selección múltiple.
* RegistroResultados.java la cual obtiene el porcentaje de respuesta desde exam.java y lo guarda en un archivo.

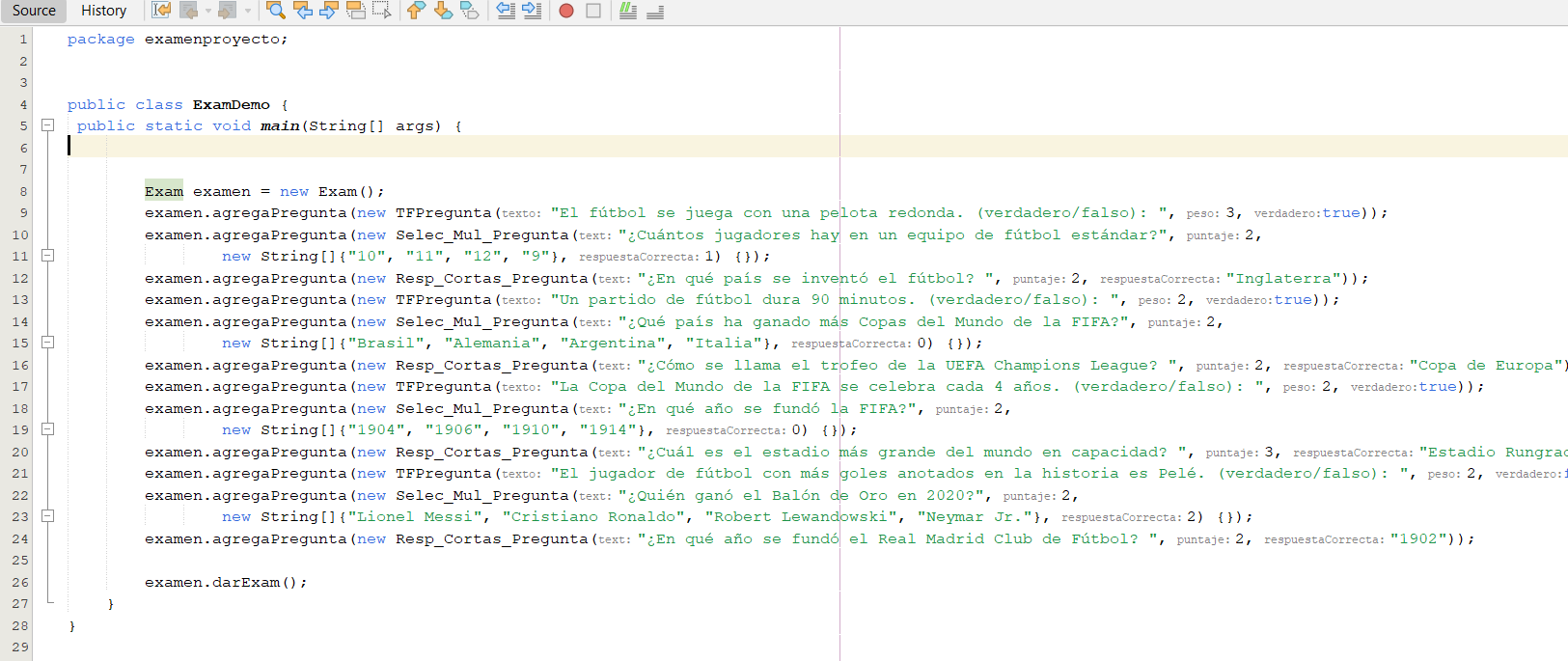
Ya cuando definimos las clases y sus funciones, comenzamos con la creación del proyecto directamente a código, nuestro tema a elección fue futbol por lo cual las preguntas son sobre eso.

Luego de algunos errores y varios testeos, nuestro código funciono, manteniendo las clases previamente descritas en el modelado del proyecto, las escribimos a detalle y serán explicadas a continuación:

ExamDemo:

La clase ExamDemo contiene el método main que ejecuta el programa. Crea un examen y agrega preguntas de futbol (nuestra elección). Luego, realiza el examen y muestra el porcentaje obtenido y el puntaje total al usuario.

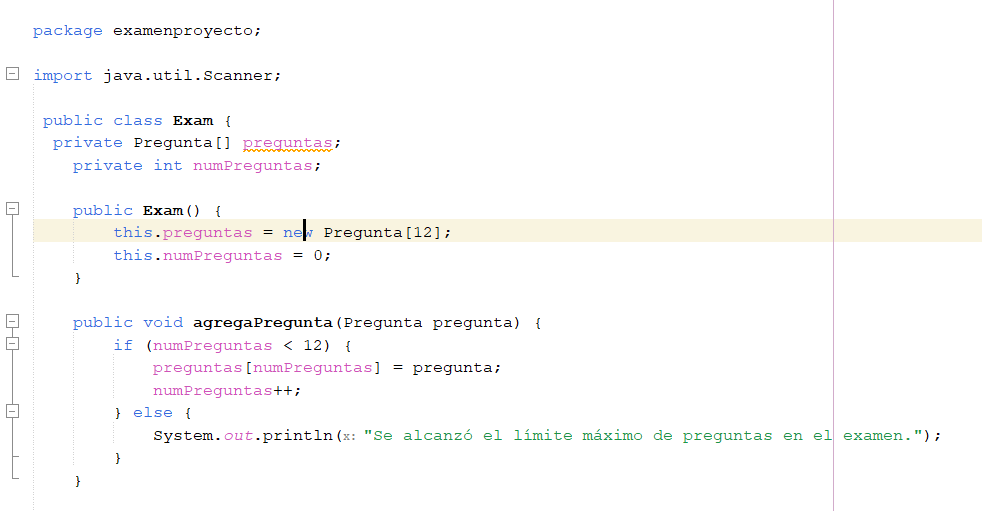
El método darExam() en el objeto examen, que inicia la ejecución del examen. El examen se ejecutará, mostrará las preguntas al usuario, recopilará las respuestas y calculará el puntaje final del usuario.

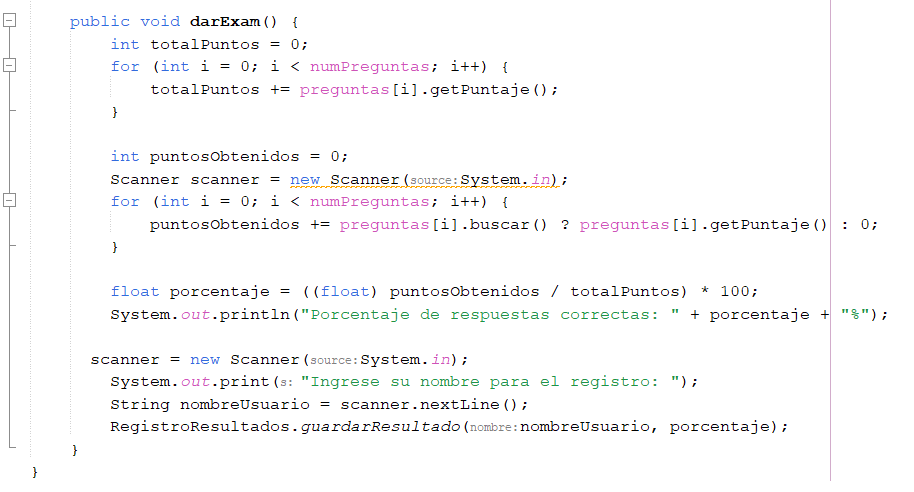


Exam:

La clase Exam administra administrar un examen, incluyendo la adición de preguntas, la ejecución del examen y la obtención de resultados. Permite agregar preguntas al examen y calcular el puntaje total y el porcentaje obtenido después de que el usuario responde a todas las preguntas. Definición de variables y métodos utilizados en esta clase:

* preguntas: Un arreglo de objetos de tipo Pregunta para almacenar las preguntas del examen.
* numPreguntas: Un entero que mantiene un registro del número de preguntas agregadas al examen.
* public Exam(), se inicializa el arreglo preguntas con capacidad para 12 preguntas y se establece numPreguntas en 0 para indicar que inicialmente no se ha agregado ninguna pregunta al examen. El metodo agregaPregunta(Pregunta pregunta) permite agregar preguntas al examen. Si el número de preguntas agregadas es menor que 12, la pregunta se agrega al arreglo preguntas y se incrementa el contador numPreguntas. Si se alcanza el límite máximo de preguntas, se muestra un mensaje de advertencia.
* darExam() se utiliza para llevar a cabo el examen.Calcula el ‘totalPuntos’ que suma los puntajes de las preguntas, inicializa ‘puntosObtenidos’ en 0 que se utiliza para realizar un seguimiento de los puntos del usuario. Utiliza un bucle para iterar las preguntas u ul¿tiliza el método buscar() en cada pregunta, si la respuesta es correcta, actualiza ‘puntosObtenidos’ y agrega puntaje de la pregunta contestada.
* Finalmente calcula el porcentaje y solicita al usuario que ingrese su nombre para registrar los resultados, llama al método ‘guardarResultados de la clase RegistroResultados para guardar el nombre y su porcentaje en el archivo.

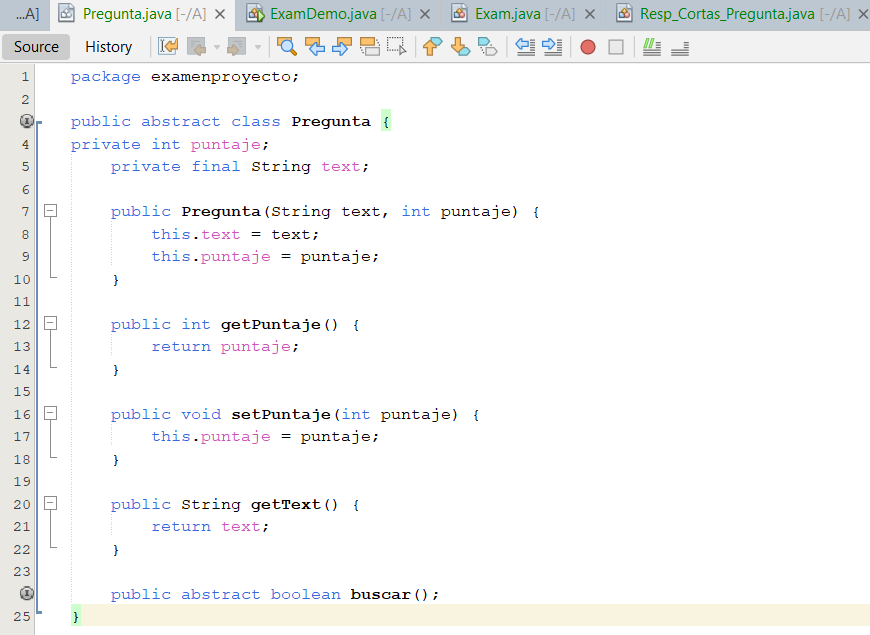




Pregunta.java:

La clase abstracta Pregunta sirve como la base para todos los tipos de preguntas en el programa. Contiene métodos para obtener el texto y el puntaje de una pregunta, así como un método abstracto buscar() para ser implementado por las clases hijas. Definimos las variables y métodos en esta clase:

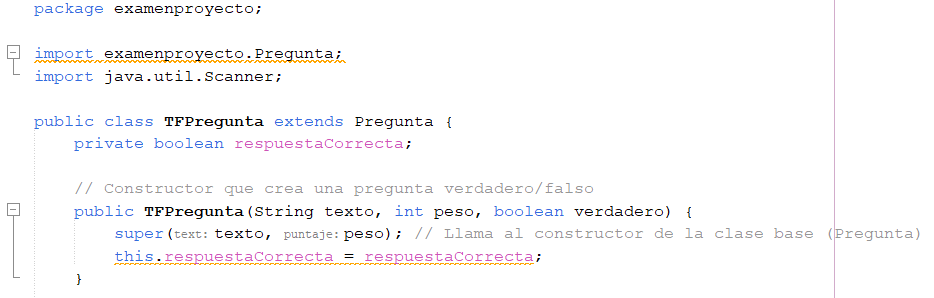
* private int puntaje, esta es una variable privada de tipo entero llamada "puntaje" que almacena la puntuación asociada con la pregunta.
* private final String text Esto es otra variable privada de tipo String llamada "text" que almacena el texto de la pregunta.
* public int getPuntaje() Este es un método público que permite obtener el valor del puntaje de la pregunta. Y public void setPuntaje(int puntaje) Este es un método público que permite establecer el valor del puntaje de la pregunta.



TFPregunta:

La clase TFPregunta extiende a Pregunta.java y representa preguntas de opción verdadero/falso en el examen. Almacena la respuesta correcta y compara la respuesta del usuario con la respuesta correcta para determinar si la respuesta es correcta o no.

@Override se utiliza antes del método buscar() para indicar que este método está sobrescribiendo el método abstracto buscar(), luego buscar() se encarga de permitir al usuario responder preguntas verdadero/falso, compara las respuesta y si coincide, muestra respuesta correcta, de lo contrario le permite al usuario intentarlo nuevamente o detener el examen.

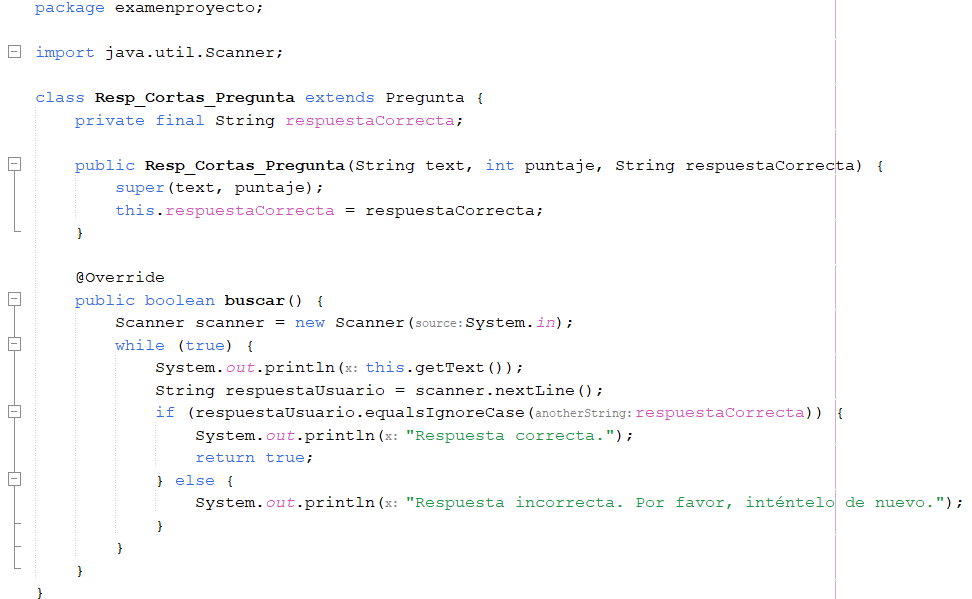




Resp\_Cortas\_Pregunta:

La clase Resp\_Cortas\_Pregunta extiende a Pregunta.java y gestiona preguntas de respuestas cortas en el examen. Compara la respuesta del usuario con la respuesta esperada, ignorando mayúsculas y minúsculas, para determinar si la respuesta es correcta.

@Override se utiliza antes del método buscar() para indicar que este método está sobrescribiendo el método abstracto buscar(), luego buscar() permite al usuario responder, el usuario ingresa su respuesta y buscar() la compara con la respuesta correcta ignorando las mayúsculas / minúsculas, si esta coincide, se devuelve “true” de lo contrario se muestra respuesta incorrecta y debe volver a intentarlo.

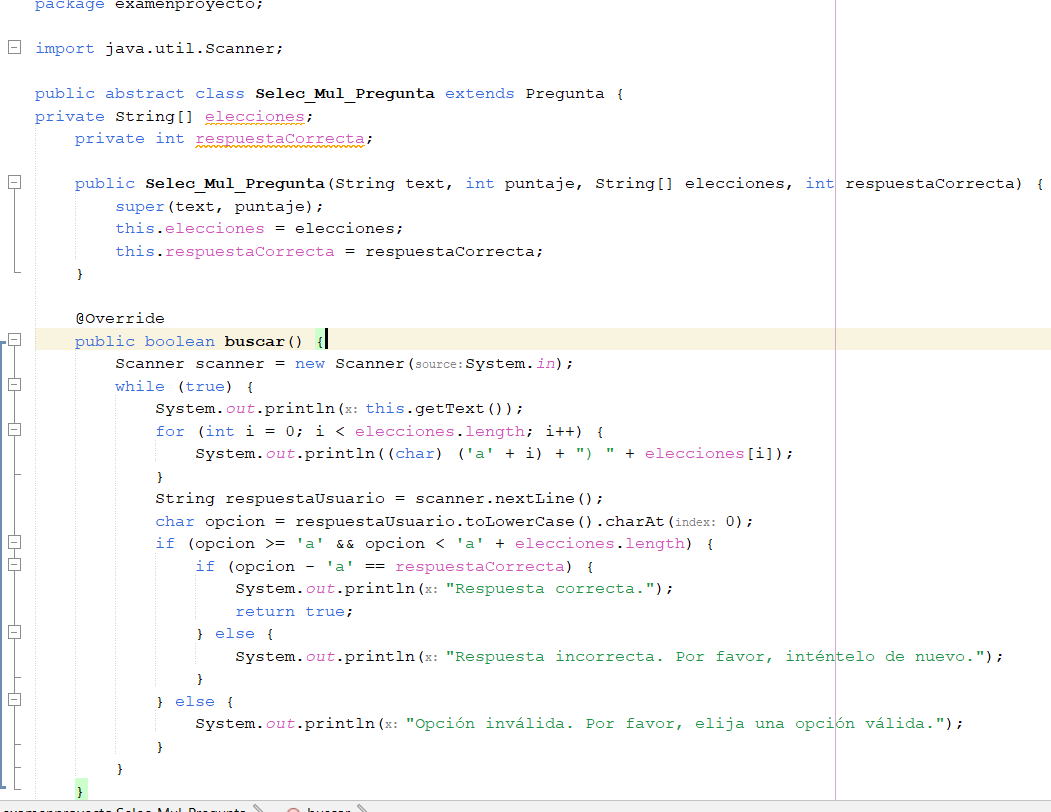


Selec\_Mul\_Pregunta:

La clase Selec\_Mul\_Pregunta extiende a Pregunta.java y maneja preguntas de opción múltiple en el examen. Permite al usuario elegir entre varias opciones y compara la respuesta seleccionada con la respuesta correcta para determinar si la respuesta es correcta. Se definen y determinan variables y métodos utilizados en esta clase:

private String[] elecciones esta es una variable miembro privada que almacena las opciones de respuesta como un arreglo de cadenas.

@Override se utiliza antes del método buscar() para indicar que este método está sobrescribiendo el método abstracto buscar(), luego buscar() muestra las opciones de respuesta(a,b,c,d…), el usuario ingresa su respuesta y se verifica si la respuesta es válida, si la opción es valida y correcta se muestra respuesta correcta y devuelve “true”, si la opción es valida de incorrecta se muestra respuesta incorrecta y permite intentarlo nuevamente y por ultimo si la opción es in valida, se muestra “Opción Invalida y le pide al usuario que ingrese una opción válida.

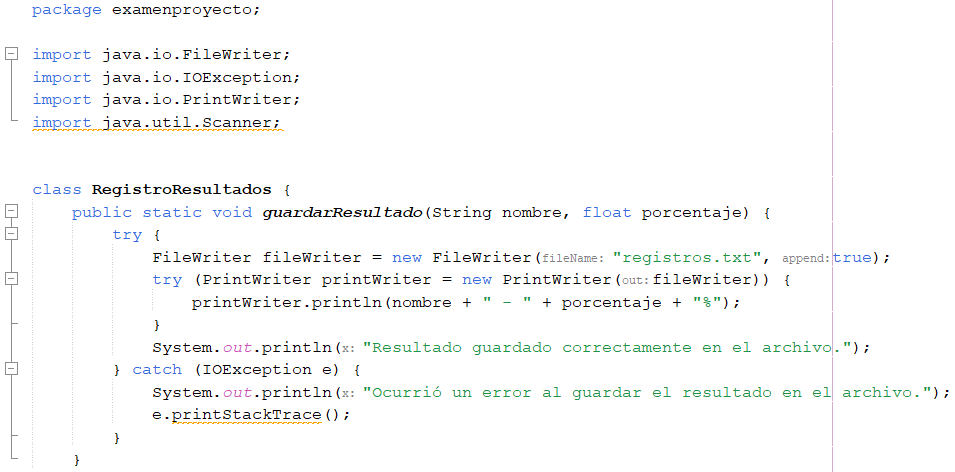


RegistroResultados:

La clase RegistroResultados guarda los resultados de un examen en un archivo de texto llamado registros.txt.

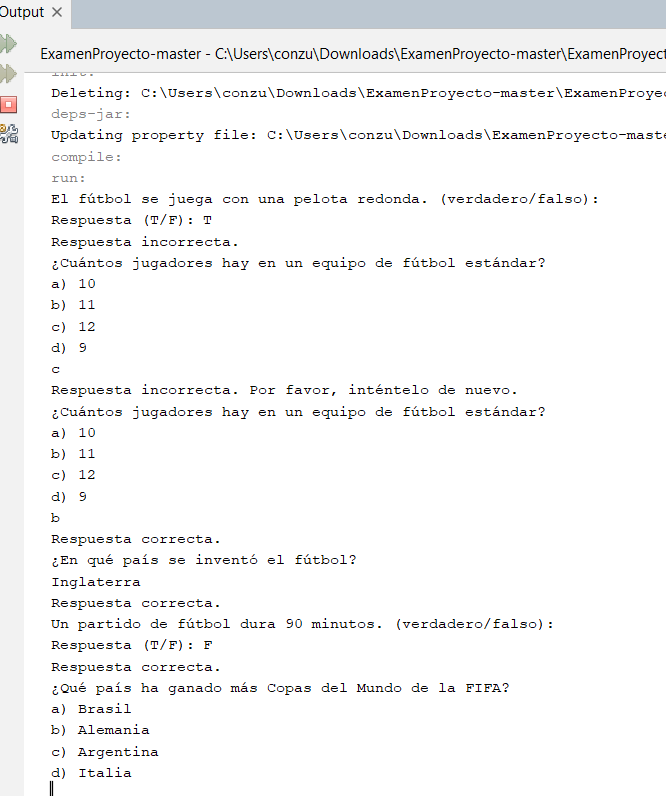
La función guardarResultado registra el nombre del usuario y el porcentaje obtenido en el examen. Si el archivo no existe, se crea automáticamente.Además utiliza un bloque "try-catch" para manejar excepciones relacionadas con la escritura de archivos. Dentro del bloque "try", se crea un objeto FileWriter para abrir el archivo "registros.txt" en modo de escritura, y se establece el segundo argumento como true, lo que significa que los datos se agregarán al final del archivo en lugar de sobrescribir el contenido existente.

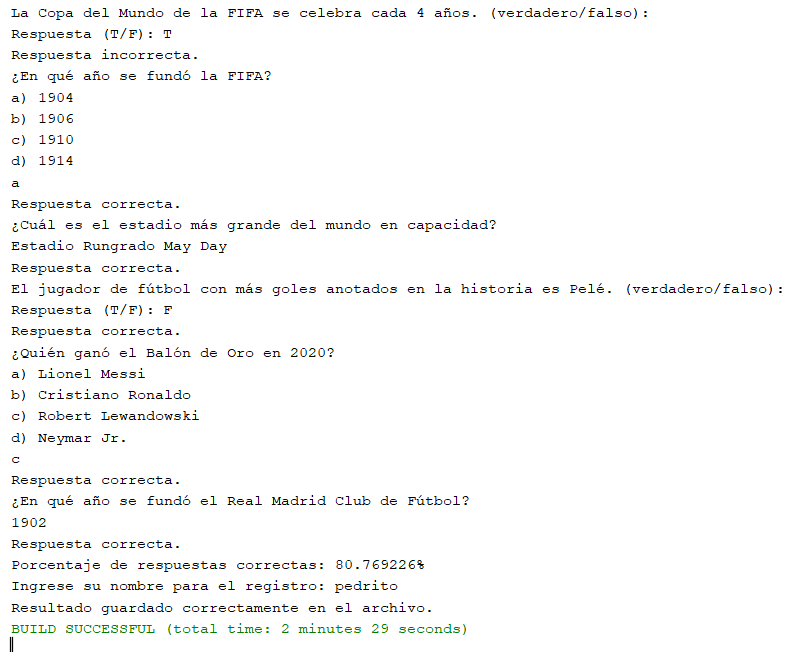
Finalmente muestra un mensaje indicando que el resultado se ha guardado correctamente en el archivo si todo se ejecuta sin errores. Si se produce una excepción de tipo IOException, se captura y se muestra un mensaje de error junto con la información de la excepción utilizando e.printStackTrace()



Ahora un ejemplo del funcionamiento por consola del proyecto.

En la imagen se puede verificar que el código tiene un buen funcionamiento y muestra todos los formatos de las preguntas he verifica el ingreso de sus respuestas, comparándolo y generando el mensaje necesario sea respuesta correcta o incorrecta.





Así concluye la presentación del proyecto, podemos decir que la creación del examen esta completa y que su funcionamiento es correcto, sus clases y la relación entre ellas funcionan de manera correcta.

Con este proyecto podemos destacar el uso de herencia para la clase pregunta y sus subclases, la correcta creación de constructores y definición de estructuras y la implementación de archivos para guardar los porcentajes y los nombres de los usuarios, junto con nuevas librerias importadas que fueron usadas para esta función como por ejemplo java.io.FileWriter y java.io.IOException, entre otros.

Para finalizar, este proyecto nos dio las herramientas necesarias para la creación de programas más completos de forma en la que entendemos los conceptos de la programación orientada a objetos y nos prepara para futuros proyectos mas complejos, el gran paso que tomamos desde la programación a estructurada a estos sistemas es grande pero muy satisfactorio ya que ahora nos facilitamos el proceso de escritura y creación del código, además de que tenemos más herramientas para su formación.