ANDROID QUIZZ APP

Cette application est un quizz qui permet de tester les connaissances de l'utilisateur sur le domaine d'Android. Le quizz comporte 20 questions. A chaque question une barre de progression affiche la progression de l'utilisateur et à chaque bonne réponse le score est incrémenté de 1. A la fin des 20 questions, une boîte de dialogue apparaît indiquant le score atteint par l'utilisateur et affiche un bouton pour fermer l'application.

Ci-dessous l'interface de l'application et l'explication de son code en grandes étapes.

Interface utilisateur :

* L'endroit où l'on se trouve dans le code sera surligné en jaune.

ETAPES	ACTIONS	EXPLICATIONS DE CODE
1	On relie les boutons true false de l'interface au fichier MainActivity.java	On crée la variable qui nous servira à manipuler les 2 boutons true et false de l'interface. On prend soin de les créer dans la MainActivity et en dehors de toute méthode ou fonction pour qu'elles soient accessibles de partout dans le code. Button trueButton ; Button falseButton ; Dans la méthode onCreate on fait le lien entre ces deux variables et les views de l'interface. On caste chaque view en (button). trueButton = (Button) findviewById(R.id.true_button) ; falseButton = (Button) findviewById(R.id.false_button) ; On applique la méthode .setOnClickListener à chacun des boutons (en utilisant une variable anonyme en paramètre de la méthode). Cela nous permet de dire ce qui va se passer quand l'un ou l'autre des boutons est pressé. `trueButton.setOnClickListener(new View.OnClickListener() {` `@Override` `public void onClick(View view) {` `//Action effectuée lorsque le bouton true est pressé` `}` `});` `falseButton.setOnClickListener(new View.OnClickListener() {` `@Override` `public void onClick(View view) {` `//Action effectuée lorsque le bouton false est pressé` `}` `});`
2	Création d'un template commun à toutes les questions	Il nous faut un template commun à toutes les questions. Pour cela on crée une autre class en dehors de MainActivity.java. Elle se nomme TrueFalse. Elle a pour attributs : -private int questionId qui contiendra le numéro de chaque question. `public class TrueFalse {` `private int questionId;` `private boolean answer;` On crée un constructeur qui prend donc le nom de la classe TrueFalse. Il prend en paramètres questionResourceId et la réponse à cette question trueOrFalse. Dans son corps le constructeur associe les paramètres à questionId et answer ; `public TrueFalse(int questionResourceId, boolean trueOrFalse){` `questionId = questionResourceId;` `answer = trueOrFalse;` `}` En dessous du constructeur on génére les getter et setter pour créer les méthodes que l'on utilisera plus tard.

		``public int getQuestionId() { return questionId; } public void setQuestionId(int questionId) { this.questionId = questionId; } public boolean isAnswer() { return answer; } public void setAnswer(boolean answer) { this.answer = answer; }```
3	Rentrer le texte des questions et la réponse juste	Dans la classe MainActivity, on crée un array qui contient toutes les questions avec leurs réponses que l'on nomme questionBank. Il est de type TrueFalse. On crée à l'intérieur de l'array chacune des 20 questions avec à chaque fois en paramètre la ressource qui nous permet de trouver la question est sa réponse. ```private TrueFalse[] questionBank = new TrueFalse[] { new TrueFalse(R.string.question_1, false), new TrueFalse(R.string.question_2, true), new TrueFalse(R.string.question_3, true), new TrueFalse(R.string.question_4, false), new TrueFalse(R.string.question_5, false), new TrueFalse(R.string.question_6, true), new TrueFalse(R.string.question_7, false), new TrueFalse(R.string.question_8, false), new TrueFalse(R.string.question_9, true), new TrueFalse(R.string.question_10, true), new TrueFalse(R.string.question_11, true), new TrueFalse(R.string.question_12, false), new TrueFalse(R.string.question_13, true), new TrueFalse(R.string.question_14, true), new TrueFalse(R.string.question_15, false), new TrueFalse(R.string.question_16, false), new TrueFalse(R.string.question_17, false), new TrueFalse(R.string.question_18, true), new TrueFalse(R.string.question_19, true), new TrueFalse(R.string.question_20, false),```
4	Changer le texte de la question qui apparaît à l'écran	On veut changer le texte qui s'affiche à l'écran pour chaque question. On doit connecter la view text_view de l'interface qui affiche la question. Dans MainActivity, on crée une variable de type TextView nommée questionTextView ```public class MainActivity extends AppCompatActivity { Button trueButton; Button falseButton; TextView questionTextView;``` Puis dans la méthode onCreate on associe la view text_view à cette variable. ```@Override protected void onCreate(Bundle savedInstanceState) { super.onCreate(savedInstanceState); setContentView(R.layout.activity_main); trueButton = (Button) findViewById(R.id.true_button);```

		`falseButton = (Button) findViewById(R.id.false_button);` `questionTextView = (TextView) findViewById(R.id.question_text_view);` Pour pouvoir afficher une question, il nous faut pouvoir avoir accès au contenu de l'array questionBank. Dans MainActivity, on crée donc une variable int index qui nous permettra d'afficher la question située à la position "index" dans l'array. `public class MainActivity extends AppCompatActivity {` `Button trueButton;` `Button falseButton;` `TextView questionTextView;` `int index;`
		Dans class MainActivity, on crée la variable question de type int qui recevra la valeur de l'id Dans la méthode onCreate, on affecte cette valeur à l'id généré automatiquement par JAVA pour l'objet contenu dans l'array questionBank à l'indice de valeur "index". Pour cela on utilise la méthode getQuestionId(). `question = questionBank[index].getQuestionId();` On utilise enfin la méthode .setText() pour afficher la question qui a l'id "question" dans la variable questionTextView `questionTextView.setText(question);`
5	changer de question l'utilisateur appuie sur les boutons true ou false	En bas de la méthode onCreate, on crée un méthode pour définir une suite d'actions standard qui devront s'effectuer lorsque l'un ou l'autre des boutons est pressé. on la nomme updateQuestion(). A chaque fois que le bouton est pressé la variable index doit augmenter de 1. On doit ensuite mettre le nouveau texte de la question dans la view questionTextView. `private void updateQuestion(){` `index = index++ ;` `question = questionBank[index].getQuestionId();` `questionTextView.setText(question);` On utilise ensuite la méthode updateQuestion() dans les deux méthodes onClicklistener des boutons true et false. `trueButton.setOnClickListener(new View.OnClickListener() {` `@Override` `public void onClick(View view) {` `updateQuestion();` `}` `});` `falseButton.setOnClickListener(new View.OnClickListener() {` `@Override` `public void onClick(View view) {` `updateQuestion();`

		`}`
6	Eviter que l'application crashe quand on arrive à la 20ème question et revenir à la première question.	Appuyer sur les boutons fait changer le texte de la question, mais arrivé à 20 l'application crashe ! Pour remédier à se problème il faut faire en sorte que l'index retombe 0 après être arrivé à 20. Pour cela, dans la méthode updateQuestion(), on utilise un mododulo sur l'incrémentation de l'index. Ainsi tant que index<20, le résultat de l'incrémentation reste inchangé, mais lorsque l'on atteint 20 il redescend à 0. Afin de pouvoir rajouter ou enlever des questions sans devoir modifier à chaque fois la valeur du modulo, on utlise la longueur de l'array qui contient les questions. `private void updateQuestion(){` `index = ++index%questionBank.length;`
7	Vérifier la réponse de l'utilisateur et l'afficher dans un message à l'utilisateur	On crée une nouvelle méthode appellée checkAnswer() en fin de méthode onCreate. Elle est privée, ne renvoie rien et prend en paramètre un boolean qui sera la réponse de l'utilisateur. A l'intérieur de cette méthode, on crée une variable de type boolean qui va récupérer la réponse correcte à la question en fonction de son indice. `private void checkAnswer(boolean userAnswer){` `bolean correctAnswer = questionBank[index].isAnswer();` `if(correctAnswer == userAnswer){` `Toast.makeText(getApplicationContext(),` `R.string.correct_toast,Toast.LENGTH_SHORT).show();` `score+=1;` `}else{` `Toast.makeText(getApplicationContext(),` `R.string.incorrect_toast,Toast.LENGTH_SHORT).show();` `}` `}` On utilise un bloc conditionnel pour afficher un message différent selon que la méthode soit juste ou non. Il faut maintenant appeler cette méthode à l'intérieur des 2 onClickListener liés aux boutons. Il faut les placer avant l'appel à la méthode updateQuestion() car sinon la méthode checkAnswer est appliquée sur la question updatée alors qu'elle doit l'être sur la question d'avant.

<table>
<tr><td>8</td><td>Faire se remplir la barre de progression à mesure que les questions avancent</td><td>Il faut mettre en relation la view qui contient la progressBar et le code JAVA. On crée donc la variable progressBar dans la classe MainActivity. Et dans la méthode onCreate() on la lie à la view progress_bar.

<pre>mProgressBar = (ProgressBar)findViewById(R.id.progress_bar);</pre>
On crée également une constante PROGRESS_BAR_INCREMENT dans la class MainActivity. La valeur pleine de la barre étant de 100, il nous faut diviser 100 par le nombre de questionS pour que la constante s'incrément correctement et soit totalement remplie à la fin du quizz. Afin d'être sûr que la barre soit totalement remplie, il faut aussi s'assurer que la valeur de la constante soit arrondie vers le haut. Le résultat étant un nombre décimal, on le caste en (int).
On écrit alors :
<pre>final int PROGRESS_BAR_INCREMENT = (int) Math.ceil (100f / questionBank.length);</pre>
ATTENTION! Ecrire la constante après l'array de questions, sinon il ne pourra pas être trouvé par la formule.</td></tr>
<tr><td></td><td></td><td>Dans méthode updateQuestion(), on peut maintenant utiliser la méthode incrementProgressBy() pour incrémenter la barre de progression.

<pre>mProgressBar.incrementProgressBy(PROGRESS_BAR_INCREMENT);</pre></td></tr>
<tr><td>9</td><td>Faire changer le score en fonction des bonnes réponses</td><td>Il faut mettre en relation la view qui contient le score et le code JAVA. On crée donc la variable score dans la classe MainActivity. Et dans la méthode onCreate() on la lie à la view score.

<pre>mScoreTextView = (TextView)findViewById(R.id.score);</pre></td></tr>
<tr><td></td><td></td><td>Dans la méthode updateQuestion() on utilise la méthode setText() pour changer le texte de la view :

<pre>mScoreTextView.setText("Score "+score+ "/"+ questionBank.length);</pre></td></tr>
<tr><td>10</td><td>Marquer la fin du quizz avec un message d'avertissement</td><td>Il nous faut faire apparaître un message lorsque l'index revient à 0.
Dans la méthode updateQuestion(), après l'incrémentation de l'index, on crée un bloc conditionnel si index est égal à 0.
On crée une boite de dialogue alert de type AlertDialog.Builder. On met en place son titre, on ulise la propriété setCancalable sur false pour que l'utilisateur ne puisse pas sortir de la boîte de dialogue en cliquant à côté. Et on affiche un message contenant le score. On utilise ensuite la propiété setPositiveButton pour définir ce qui se passe quand l'utilisateur clique sur le bouton de la boite de dialogue. On utlise également la méthode .show() pour faire apparaître le message à l'écran.

<pre>if (index == 0) {
    AlertDialog.Builder alert = new AlertDialog.Builder(this);
    alert.setTitle("Game Over");
    alert.setCancelable(false);</pre></td></tr>
</table>

		`` alert.setMessage("You scored " + score + " points!"); alert.setPositiveButton("Close Application", new DialogInterface.OnClickListener() { @Override public void onClick(DialogInterface dialogInterface, int i) { finish(); } }); alert.show(); } ```
11	Garder les variables updatées lorsque l'on tourne l'écran en mode landscape	Lorsque l'on tourne l'écran, l'application est réinitialisée. Pour éviter cela, il nous faut overrider la methode onSaveInstanceState() en fin de MainActivity. Elle prend en pamètre un objet de type Bundle appelé outState. Cela nous permet de conserver l'état de notre application lorsque l'écran est tourné. On doit conserver les valeurs pour l'index et le score. ``` @Override public void onSaveInstanceState(Bundle outState){ super.onSaveInstanceState(outState); outState.putInt("ScoreKey", score); outState.putInt("IndexKey", index); } } ``` Maintenant il nous faut vérifier lorsque la méthode onCreate() est activée si savedInstanceState est null ou pas, c'est à dire si l'utilisateur a déjà répondu à des questions et si oui conserver son score et l'index de la question à laquelle il se trouve.Si savedInstanceState est null les varaible score et index sont égales à 0. ``` @Override protected void onCreate(Bundle savedInstanceState) { super.onCreate(savedInstanceState); setContentView(R.layout.activity_main); if (savedInstanceState != null){ score = savedInstanceState.getInt("ScoreKey"); index = savedInstanceState.getInt("IndexKey"); }else{ score = 0; index = 0; } ``` Il nous faut maintenant updater le score mais en le plaçant dans le code après que la variable scoreTextView se soit déjà vue assigner une valeur. ``` questionTextView = (TextView) findViewById(R.id.question_text_view); mScoreTextView = (TextView) findViewById(R.id.score); mProgressBar = (ProgressBar) findViewById(R.id.progress_bar); question = questionBank[index].getQuestionId(); questionTextView.setText(question); //on peut ici changer le contenu de scoreTextView car il a déjà une valeur mScoreTextView.setText("Score "+score+ "/"+ questionBank.length); ```