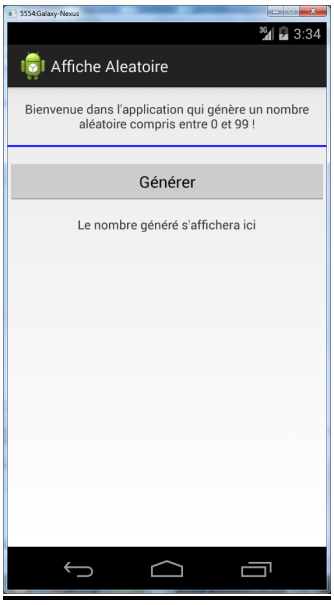
Gestion de l'interactivité - Boutons, zones de saisie ...

Exercice 1 - Génération d'un nombre aléatoire

L'inconvénient de l'application précédente est que si l'on veut générer plusieurs nombres aléatoires, il faut la relancer.

Le but de l'exercice est de développer une application permettant de générer des nombres aléatoires. L'interface graphique sera dotée d'un bouton. Chaque fois que l'utilisateur appuiera sur le bouton, un nouveau nombre aléatoire sera généré.

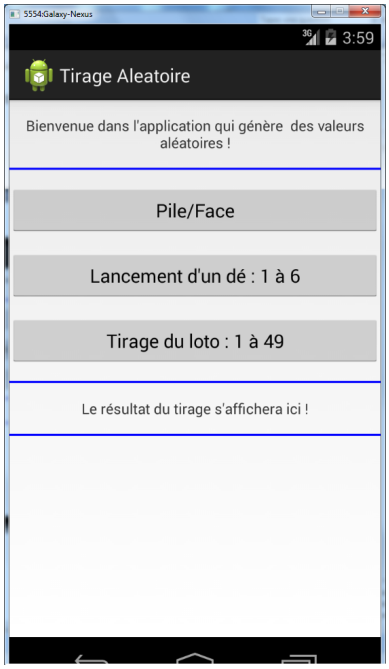


Exercice 2 - Différents tirages aléatoires

On souhaite spécialiser l'application précédente pour qu'elle permette d'effectuer des tirages aléatoires précis :

- ✓ un tirage à pile ou face
- ✓ simuler le lancement d'un dé, donc tirage d'un chiffre entre 1 et 6
- ✓ tirage d'un numéro pour le loto, un entier compris entre 1 et 49

On prévoira donc dans l'application 3 boutons différents permettant d'obtenir les 3 tirages différents (voir des exemples, page suivante)

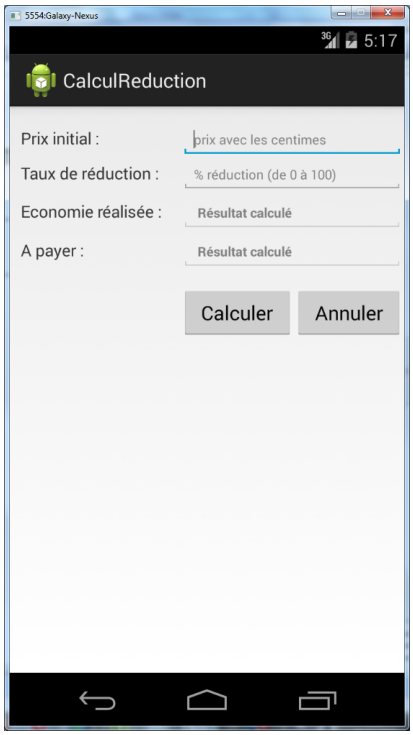


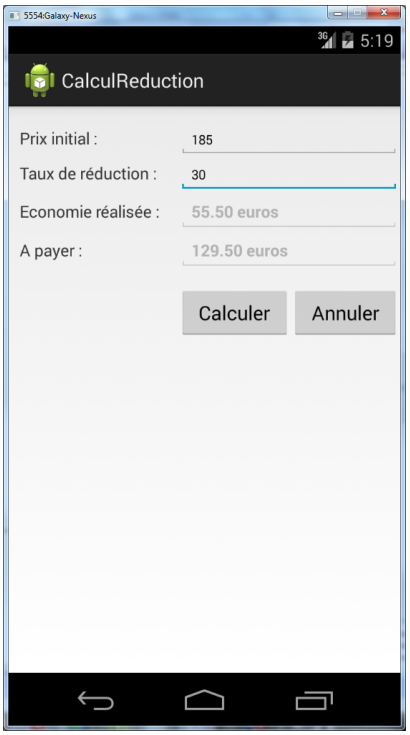
1) Développer l'application ainsi décrite.

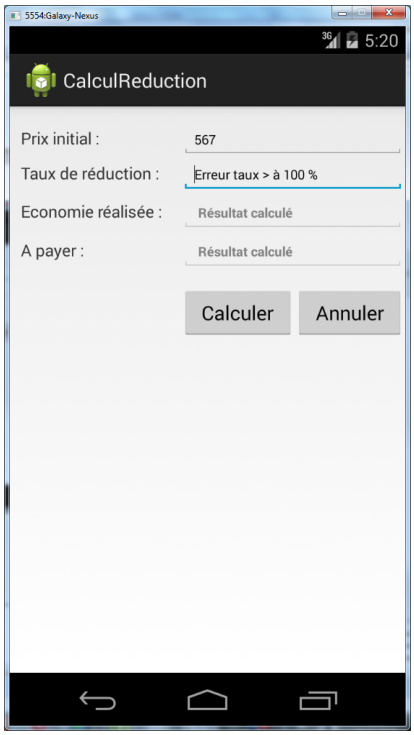
2) Facultatif : Améliorer l'interface en utilisant des couleurs pour les textes affichés par les *TextView*, et pour le fond des boutons.

Exercice 3 - Calcul du prix à payer après une réduction

Le but de ce TP est de développer une application permettant de calculer le prix à payer pour un article sur lequel une réduction s'applique.







Dans une première zone de saisie, l'utilisateur entrera le prix avant la réduction. Ce prix sera exprimé en euros, avec éventuellement des centimes d'euro. Dans la deuxième zone de saisie, l'utilisateur entrera le taux de réduction. Avant qu'il ne saisisse ces valeurs, une indication sera affichée dans ces deux zones, conformément à l'exemple ci-dessus.

Lorsque l'utilisateur appuiera sur le bouton "Calculer", le montant à payer s'affichera dans la zone de résultat, une ligne indiquera également l'économie réalisée.

A tout moment, l'utilisateur pourra appuyer sur le bouton "Annuler" pour réinitialiser les champs avec les valeurs par défaut. L'utilisateur ne doit pas avoir la possibilité d'entrer une valeur dans les zones de résultat.

a) Réaliser le traitement demandé, dans un premier temps sans contrôler la validité des valeurs saisies. On utilisera le gestionnaire de mise en forme `TableLayout`.

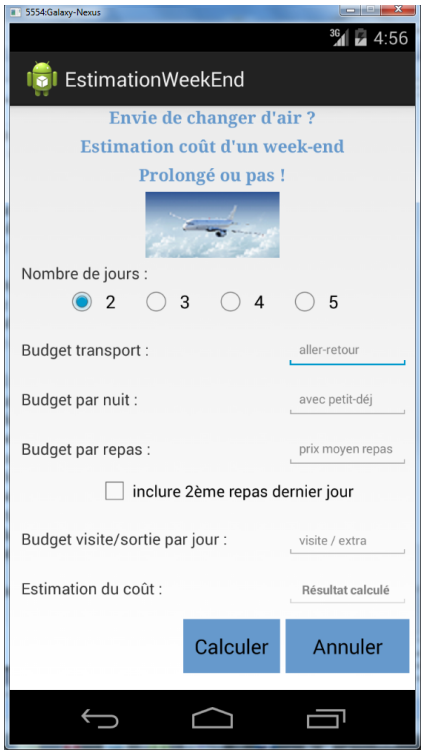
b) On impose maintenant que les deux valeurs numériques soient positives ou nulles. De plus, le taux de réduction devra être égal au plus à 100. On respectera les contraintes suivantes :

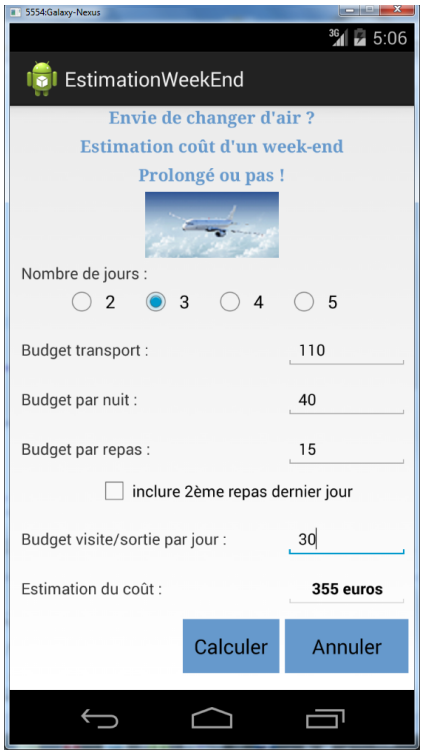
- ✓ empêcher l'utilisateur de saisir une valeur négative dans les zones de saisie
- ✓ si l'utilisateur entre un taux supérieur à 100, un message apparaîtra dans cette zone de saisie pour l'informer de son erreur : *"Erreur, taux > à 100 %"*
- ✓ si l'utilisateur n'entre pas de valeur, ou entre une seule des deux valeurs demandées, l'application ne devra pas s'arrêter en état d'erreur.

c) Améliorer l'interface en utilisant des couleurs pour les textes affichés par les *TextView*, pour le fond des boutons. Eventuellement ajouter une image de votre choix qui sera affichée en dessous des boutons.

Exercice 4 - Estimation d'un coût d'un week-end

On souhaite développer une application pour estimer rapidement le coût d'un week-end, prolongé ou pas. Au lancement de l'application, l'interface sera dans l'état ci-dessous, à gauche.





Par défaut, le nombre de jours du week-end sera égal à 2. Une indication sera affichée dans chaque zone de saisie. L'utilisateur sélectionnera le nombre de jours de son week-end prolongé ou pas : un nombre compris entre 2 et 5. Il indiquera ensuite le budget prévu pour :

- le transport
- chaque nuit y compris le petit-déjeuner
- chaque repas
- les visites, ou loisirs de chaque jour.

On veillera à ce que l'utilisateur ne puisse saisir que des nombres entiers positifs ou nuls. L'idéal est de bloquer la saisie de tout caractère autre qu'un chiffre. A défaut, si l'utilisateur a la possibilité de saisir des caractères autres que des chiffres, un clic sur le bouton *Calculer* doit rétablir la vue dans son état initial.

Un clic sur le bouton *Annuler* doit rétablir l'affichage dans son état initial.

Un clic sur le bouton *Calculer* provoquera l'affichage du coût estimé. Les règles de calcul sont les suivantes :

- Le nombre de nuits est égal au nombre de jours moins 1
- Il faut compter 2 repas par jour, sauf pour le dernier jour où un seul repas est compté. L'utilisateur a la possibilité de rajouter ce dernier repas en cochant la case prévue à cet effet
- Si un tarif n'est pas renseigné, il sera considéré comme égal à 0.

Contraintes à respecter

- Une photo sera affichée en dessous des 3 lignes d'introduction. Le contenu de la photo est libre.

- L'interface sera développée en utilisant le gestionnaire de mise en forme `TableLayout`.

- Il sera nécessaire de gérer correctement et uniformément différentes marges ou taille de caractères. Les valeurs de celles-ci n'apparaîtront pas explicitement dans le fichier *xml* décrivant la vue. Mais elles seront définies dans un fichier de ressources nommé ***dimensions.xml***.

- On définira des couleurs dans un fichier adéquat. Eventuellement, des couleurs supplémentaires pourront être ajoutées à l'interface.

- On définira au moins 3 styles dans le fichier ***styles.xml***. L'un d'entre-eux permettra de décrire le style des 3 lignes d'introduction de l'application : la couleur, la police, la taille des caractères, leur style, leur positionnement (centré), la marge entre les lignes...

Exercice 5 - Calcul du prix à payer après une réduction

Refaire l'exercice 3 en utilisant le gestionnaire de mise en forme `RelativeLayout`.