Informatique Mobile, Ubiquitaire et Persuasive + Kotlin, Interactions et États, Activités

8INF865

UQAC
Université du Québec
à Chicoutimi

Ordre du jour

- Kotlin
- Activité (proto-persona)
- Historique des (changements de) paradigmes
- Informatique Mobile
- Informatique Ubiquitaire
- Informatique Persuasive
- Activité (flux d'utilisation)
- Interaction et états (Jetpack Compose)
- Activité (croquis)
- Liste et Style (Material Design) - Exercices

Kotlin

Generics
Enum class
Data class
Object (et Companion)
Extensions
Interface
Scope functions
Collections
High-order functions

Generics

```
class Question<T>(
    val questionText: String,
    val answer: T,
    val difficulty: String
)

fun main() {
    val q1 = Question<String>(
        "Quoth the raven ___",
        "nevermore",
        "medium"
        )
    val q2 = Question<Boolean>("The sky is green. True or false", false, "easy")
    val q3 = Question<Int>("How many days between full moons?", 28, "hard")
}
```

```
class [class name] < [generic data type] >(
    val [property name] : [generic data type]
)
```

```
val [instance name] = [class name] < [generic data type] >( [parameters] )
```

Enum class

```
enum class Difficulty {
    EASY, MEDIUM, HARD
}
class Question<T>(
    val questionText: String,
    val answer: T,
    val difficulty: Difficulty
)
val q1 = Question<String>(
    "Quoth the raven ___",
    "nevermore",
    Difficulty.MEDIUM
    )
val q2 = Question<Boolean>("The sky is green. True or false", false,   Difficulty.EASY)
val q3 = Question<Int>("How many days between full moons?", 28,   Difficulty.HARD)
```

```
enum class enum name {
    Case 1 , Case 2 , Case 3

}
```

enum name . case name

Data class

```
data class Question<T>(
    val questionText: String,
    val answer: T,
    val difficulty: Difficulty
)
```

Le constructeur d'une classe de données doit comporter au moins un paramètre. Les paramètres du constructeur doivent être marqués en tant que `val` ou `var`

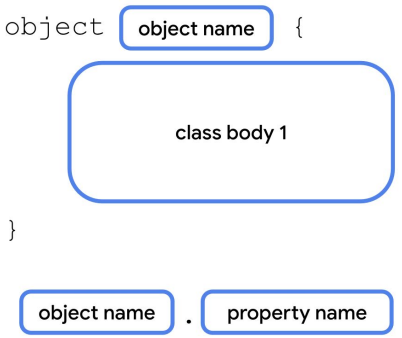
```
data class class name ( ... )
```

Méthodes implémentés

- equals()
- hashCode(): you'll see this method when working with certain collection types.
- toString()
- componentN(): component1(), component2(), etc.
- copy()

Singleton (object)

```
object QProgress {
    var total: Int = 10
    var answered: Int = 3
}


fun main() {
    ...
    println("${QProgress.answered} of ${QProgress.total} answered.")
}
```



Companion (object)

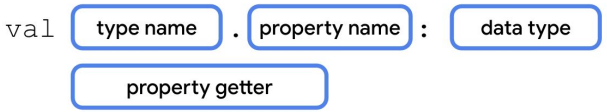
```
class Quiz {
    val q1 = Question<String>("Quoth the raven ___", "nevermore", Difficulty.MEDIUM)
    val q2 = Question<Boolean>("The sky is green. True or false", false, Difficulty.EASY)
    val q3 = Question<Int>("How many days between full moons?", 28, Difficulty.HARD)

    companion object QProgress {
        var total: Int = 10
        var answered: Int = 3
    }
}

fun main() {
    ...
    println("${Quiz.answered} of ${Quiz.total} answered.")
}
```

companion object **object name**

Extension (property)

```
padding(16.dp)
```

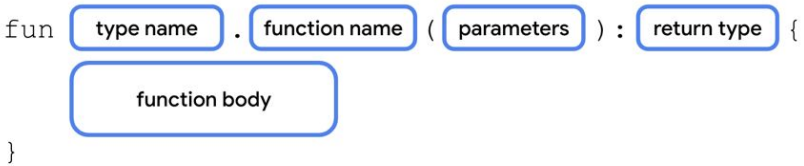
```
class Quiz {
    ...
    companion object QProgress {
        ...
    }
}

val Quiz.QProgress.progressText: String
    get() = "${answered} of ${total} answered"

fun main() {
    println(Quiz.progressText)
}
```



L'extension (propriété) ne peut pas stocker de données (donc uniquement de type "get")

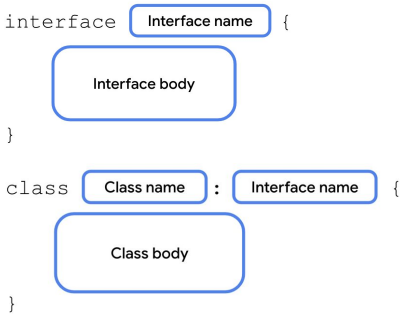
Extension (function)

```
class Quiz {
    ...
    companion object QProgress {
        ...
    }
}

fun Quiz.StudentProgress.printProgressBar() {
    ...
    println(Quiz.progressText)
}

fun main() {
    Quiz.printProgressBar()
}
```



Interface

```
interface ProgressPrintable {
    val progressText: String
    fun printProgressBar()
}

class Quiz : ProgressPrintable {
    ...
    override fun printProgressBar() {
        ...
    }
}

fun main() {
    Quiz().printProgressBar()
}
```



Scope functions

`let()`

Pour remplacer les longs noms d'objets

`apply()`

Pour appeler des méthodes d'un objet sans variable

```
fun printQuiz() {
    println(question1.questionText)
    println(question1.answer)
    println(question1.difficulty)
    println()
    println(question2.questionText)
    println(question2.answer)
    println(question2.difficulty)
    println()
    println(question3.questionText)
    println(question3.answer)
    println(question3.difficulty)
    println()
}
```

```
fun main() {
    val quiz = Quiz()
    quiz.printQuiz()
}
```

```
fun printQuiz() {
    question1.let {
        println(it.questionText)
        println(it.answer)
        println(it.difficulty)
    }
    println()
    question2.let {
        println(it.questionText)
        println(it.answer)
        println(it.difficulty)
    }
    println()
    question3.let {
        println(it.questionText)
        println(it.answer)
        println(it.difficulty)
    }
    println()
}
```

```
fun main() {
    Quiz().apply {
        printQuiz()
    }
}
```

Array

Taille fixe

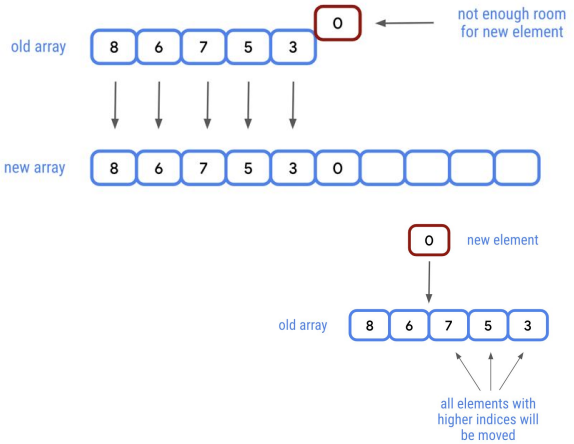
```
val variable name = arrayOf<data type>(element1, element2, ...)
```

```
val rockPlanets = arrayOf<String>("Mercury", "Venus", "Earth",
"Mars")
val gasPlanets = arrayOf("Jupiter", "Saturn", "Uranus", "Neptune")

val solarSystem = rockPlanets + gasPlanets

println(solarSystem[0])

solarSystem[3] = "Little Earth"
```

```
array name [ index ]

array name [ index ] = value
```

List, MutableList

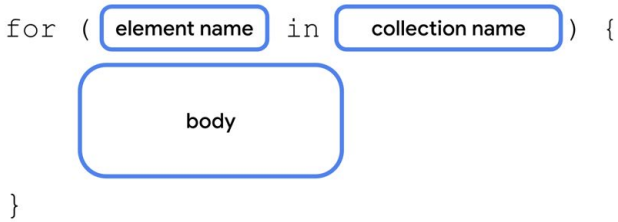
```
val solarSystem = listOf("Mercury", "Venus", "Earth")
println(solarSystem.size)
println(solarSystem[2])
println(solarSystem.get(2))
println(solarSystem.indexOf("Earth"))
solarSystem.add("Pluto")
solarSystem.add(1, "Theia")
solarSystem[2] = "Future Moon"
solarSystem.removeAt(1)
solarSystem.remove("Future Moon")
println(solarSystem.contains("Pluto"))
println("Future Moon" in solarSystem)
```



Iteration (for)

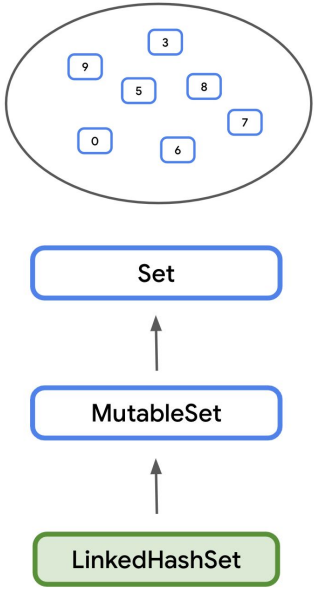
```
val solarSystem = listOf("Mercury", "Venus", "Earth", "Mars", "Jupiter")
```

```
for (planet in solarSystem) {
    println(planet)
}
```



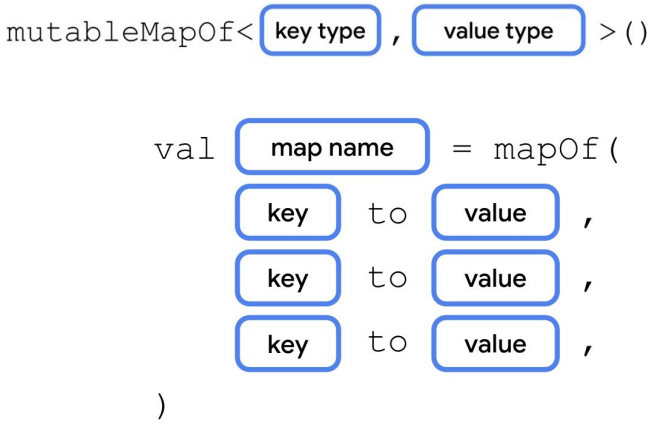
Set, MutableSet

```
val solarSystem = mutableSetOf("Mercury", "Venus", "Earth",
"Mars")

println(solarSystem.size)
solarSystem.add("Pluto")
println(solarSystem.contains("Pluto"))
solarSystem.remove("Pluto")
```



Map, MutableMap

```
val solarSystem = mutableMapOf(
    "Mercury" to 6,
    "Mars" to 2,
    "Jupiter" to 79,
    "Saturn" to 82
)
println(solarSystem.size)
solarSystem["Pluto"] = 5
println(solarSystem["Pluto"])
println(solarSystem.get("Theia"))
solarSystem.remove("Pluto")
```



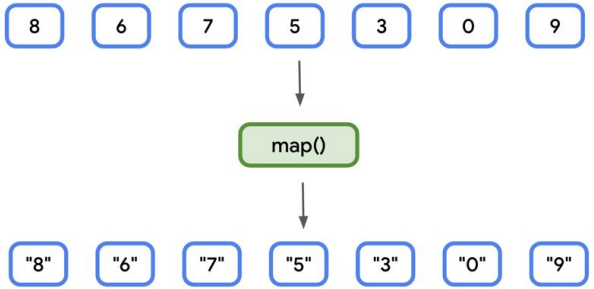
High-order functions **(forEach)**

```
class Cookie(
    val name: String,
    val price: Double
)

val cookies = listOf(
    Cookie(name = "Chocolate Chip", price = 1.69),
    Cookie(name = "Banana Walnut", price = 1.49),
)

fun main() {
    cookies.forEach {
        println("Menu item: ${it.name}")
    }
}
```

"${ expression }"

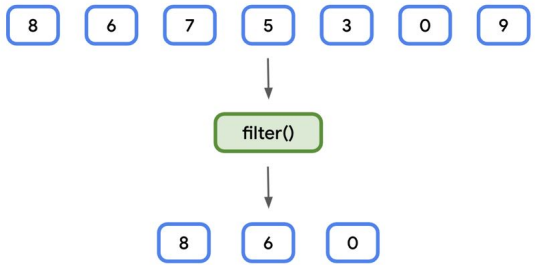
High-order functions **(map)**

```
class Cookie(
    val name: String,
    val price: Double
)

val cookies = listOf(
    Cookie(name = "Chocolate Chip", price = 1.69),
    Cookie(name = "Banana Walnut", price = 1.49),
)

fun main() {
    val fullMenu = cookies.map {
        "${it.name} - $${it.price}"
    }
}
```



High-order functions **(filter)**

```
class Cookie(
    val name: String,
    val softBaked: Boolean
)

val cookies = listOf(
    Cookie(name = "Chocolate Chip", softBaked = false),
    Cookie(name = "Banana Walnut", softBaked = true),
)

fun main() {
    val softBakedMenu = cookies.filter {
        it.softBaked
    }
}
```

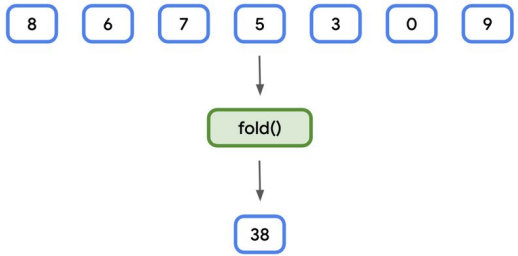


High-order functions **(groupBy)**

```
class Cookie(
    val name: String,
    val softBaked: Boolean
)

val cookies = listOf(
    Cookie(name = "Chocolate Chip", softBaked = false),
    Cookie(name = "Banana Walnut", softBaked = true),
)

fun main() {
    val softBakedMenu = cookies.groupBy {
        it.softBaked
    }
    val softBakedMenu = groupedMenu[true] ?: listOf()
    val crunchyMenu = groupedMenu[false] ?: listOf()
}
```

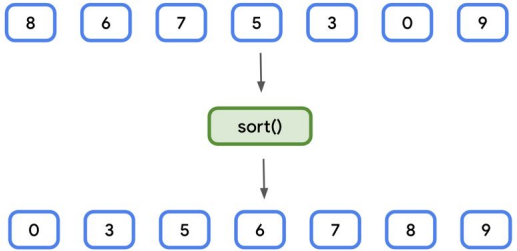
High-order functions **(fold)**

```
class Cookie(
    val name: String,
    val price: Double
)

val cookies = listOf(
    Cookie(name = "Chocolate Chip", price = 1.69),
    Cookie(name = "Banana Walnut", price = 1.49),
)

fun main() {
    val totalPrice = cookies.fold(0.0) {total, cookie ->
        total + cookie.price
    }
}
```



High-order functions **(sortedBy)**

```
class Cookie(
    val name: String,
    val price: Double
)

val cookies = listOf(
    Cookie(name = "Chocolate Chip", price = 1.69),
    Cookie(name = "Banana Walnut", price = 1.49),
)

fun main() {
    val alphabeticalMenu = cookies.sortedBy {
        it.name
    }
}
```



Activité
Proto-persona

Activité
(20 minutes)

Proto-persona

Votre tâche

1. **Individuellement** (~ 10 minutes)
    a. Créez une proto-persona en lien avec l'idée, le thème, le concept de votre application de projet

2. **En équipe** (~ 10 minutes)
    a. Présentez vos proto-personas
    b. Identifiez les éléments que vos proto-personas ont en commun et ceux qui sont différents
    c. Éliminez, combinez ou modifiez de manière à générer 2 personas, qui représentent des groupes de personnes utilisatrices aux objectifs, frustrations et comportements distincts

Nom
Occupation
Information démographique
Intérêts
Personnalité…

Objectifs

Quels sont les objectifs de … en lien avec … et comment ils s’imbriquent dans sa vie de tous les jours

Frustrations

Quels sont les points qui frustrent … en lien avec le domaine d’application

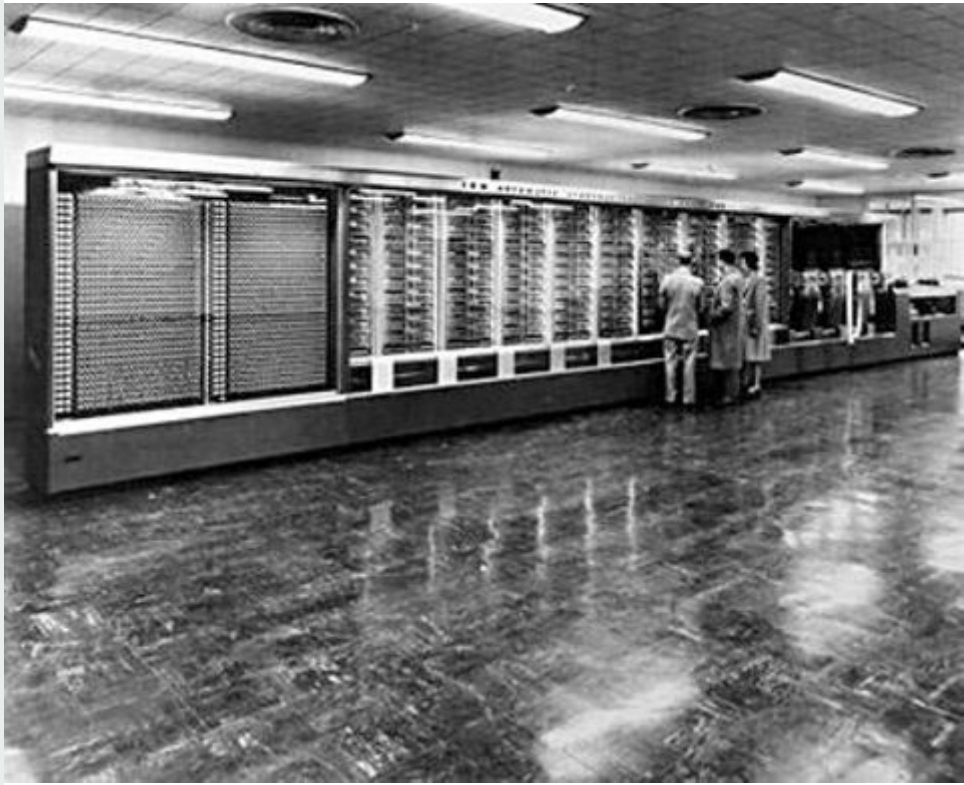
Comportements

Que fait … dans le contexte dans le domaine d’application (utilisation des technologies, fréquence…)

Historique des (changements de) **paradigmes**

Mainframe

1 machine
N personnes utilisatrices



Harvard Mark

https://www.computerhistory.org/timeline/1944

Personal Computer
(PC)

1 machine
N personnes utilisatrices
(plus petit N)

Personal Computer (PC) **Portable**

1 machine
1 personne utilisatrice

Laptop

https://commons.wikimedia.org/wiki/File:Desk-laptop-notebook-table_(23697249344).jpg

Informatique Mobile

(aujourd'hui)

N machines
1 personne utilisatrice

Weber, Dominik & Voit, Alexandra & Kratzer, Philipp & Henze, Niels. (2016). ***In-situ investigation of notifications in multi-device environments****. 1259-1264. 10.1145/2971648.2971732.*

Informatique Ubiquitaire

(présent-futur)

N machines
N personnes utilisatrices

https://news.mit.edu/2020/iot-deep-learning-1113

Human-Computer Integration

(futur)

N machines
1 utilisateur

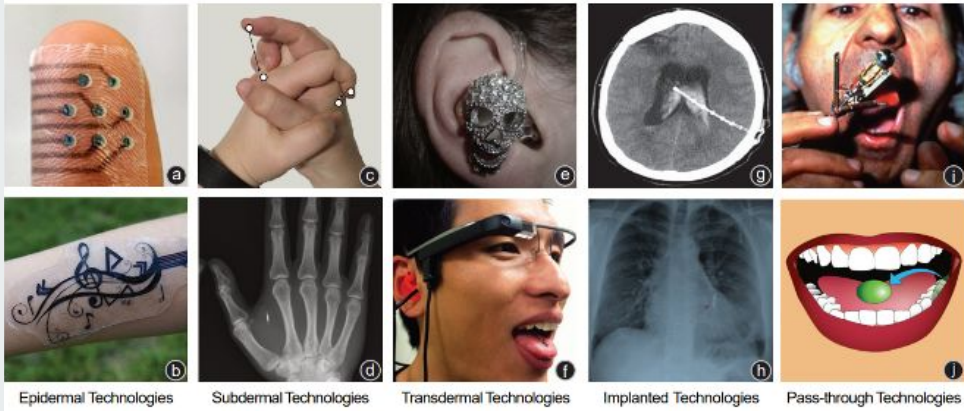


Figure 3. Examples of human-compatible technologies. (a) Tacttoo [98], (b) iSkin [95], (c) Hobbyist use of insertable devices [28], (d) RFID implants [25] (e) Wear It Loud [65], (f) The tongue and ear interface [69], (g) Cerebral shunts [77], (h) pacemaker [88], (i) Stomach Sculpture [82], (j) ChewIt [23].

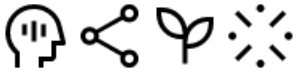
Mueller et al., 2020. Next Steps for Human-Computer Integration. In Proceedings of the 2020 CHI Conference on Human Factors in Computing Systems (CHI '20). Association for Computing Machinery, New York, NY, USA, 1–15. https://doi.org/10.1145/3313831.3376242

Informatique Mobile

avec [accompagne] la personne utilisatrice

Informatique Ubiquitaire

partout, en tout temps, à l'arrière-plan



Informatique Persuasive

modification du comportement

Informatique mobile

Importance des appareils mobiles

Impact la vie de nombreuses personnes

- Divertissement
  - musique, vidéo, jeux, etc.
- Accès constant à l'information
  - recherche et navigation
- Partage de contenu multimédia (en direct ou asynchrone)
  - communication, publications

Actions principales

- Communication (messagerie, courriels)
- Jeux
- Réseaux sociaux
- Multimédia (vidéo, musique)
- Recherche d'information
- Achats
- Navigation
- Lecture

Muller et al.(2012)

Unicité des technologies mobiles

Média capture

- Capture des moments, souvenirs
- Possibilité de partager sa vie avec les autres

Capteurs

- Localisation, accéléromètre, compas, senseur corporel
- Suivre sa santé personnelle

Connectivité

- Périphérique toujours avec une personne
- Voix, texte, email, chat, partage de médias

Appareils mobiles - statistiques globales

Année	Nombre de téléphones intelligents	Nombre de personnes utilisatrices de téléphones intelligents
2029*	8.06	6.38
2028*	7.95	6.22
2027*	7.77	6.01
2026*	7.58	5.65
2025*	7.43	5.28
2024	**7.21**	**4.88**

**Chiffre estimatif selon Ericsson*

https://www.bankmycell.com/blog/how-many-phones-are-in-the-world

86%

de la population terrestre

Appareils mobiles et internet

« D'ici 2025, **72%** de tous les utilisateurs de l'internet utiliseront **uniquement des appareils mobiles** »

→ Mentalité d'affaire « *mobile-first* »

https://www.bankmycell.com/blog/how-many-phones-are-in-the-world

Caractéristiques dédiées

Contraintes visuelles

- Éviter la surcharge
- Texte et boutons trop petits
- Contraste faible

Interactions incertaines

- Interactions gestuelles
- Fonctionnalités cachées

Expérience multi-appareils

- Disponibilité de l'information
- Interactions adaptées

Contexte de l'utilisateur

- Multitasking
- Environnement
- Distractions et interruptions

Finalement, qu'est-ce que c'est ? *(l'informatique mobile)*

La branche d'étude de l'informatique se concentrant sur les interactions entre l'utilisateur et un ou des appareil(s) **mobile(s)** .

Ces appareils et interactions tiennent compte de la **portabilité** des appareils, de leurs capacités de **communication** ainsi que du **contexte** changeant de l'utilisateur.

Portabilité et connectivité

Ordinateur portable, tablette, téléphone intelligent, montre connectée

- Petite taille
- Alimentation par batterie

Communication

- mobile – serveur
- mobile – mobile
- mobile – périphérique
- mobile – objets

Contexte (1/2)

« Le contexte est toute information qui peut être utilisée pour **caractériser la situation** d'une entité. Une entité est une personne, un lieu ou un objet considéré comme pertinent pour l'interaction entre un utilisateur et une application, y compris l'utilisateur et les applications elles-mêmes »

- *traduit librement de [Dey et al. 2001]*

Contexte (2/2)

Pourquoi s'y intéresser ?

Réduire la charge mentale de l'utilisateur

- Adaptations, fonctionnalités proactives
- Recherche et filtres d'informations
- Contrôle des interruptions
- Environnements intelligents

Contexte. Capteurs

Physique

Fait référence à la mesure d'un phénomène/mesurande physique

ex., location, lumière, son, mouvement, touché, température, rythme cardiaque, etc.

Virtuel

Information accessible dans l'appareil, mais ne faisant pas référence à un phénomène/mesurande physique

ex., application à l'avant plan, temps depuis le dernier unlock, etc.

Contexte.
Capteurs physiques

Type of context	*Available sensors*
Light	Photodiodes, colour sensors, IR and UV-sensors etc.
Visual context	Various cameras
Audio	Microphones
Motion, acceleration	Mercury switches, angular sensors, accelerometers, motion detectors, magnetic fields
Location	Outdoor: Global Positioning System (GPS), Global System for Mobile Communications (GSM); Indoor: Active Badge system, etc.
Touch	Touch sensors implemented in mobile devices
Temperature	Thermometers
Physical attributes	Biosensors to measure skin resistance, blood pressure

Contexte. Capteurs virtuels

Souvent déterminé à partir d'information disponible par rapport à l'état de l'appareil
→ *ne nécessite pas la mesure d'un phénomène/mesurande physique*

Par exemple :

- heure
- écran ouvert/éteint
- temps depuis le dernier redémarrage

Contexte. Capteurs logiques

Souvent déterminé à partir de :

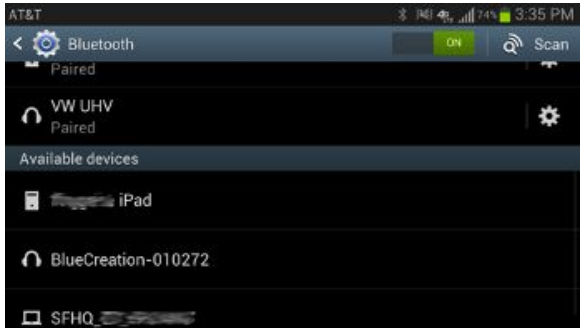
- fusions de mesures de capteurs physiques et/ou virtuels
- l'utilisation de modèle d'apprentissage machine/d'intelligence artificielle pour inférer de l'information sur le contexte utilisateur

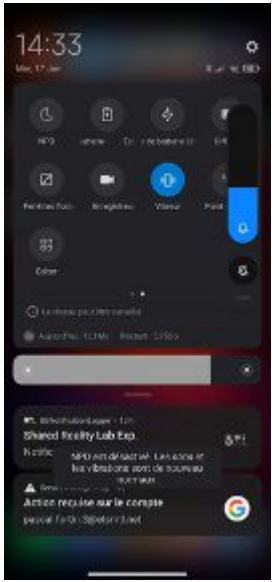
Par exemple :

- activité (reconnue à partir des capteurs de mouvements)
- tâche (inféré à partir des applications ouvertes)
- état émotionnel (estimé à partir de mesures physiologiques et comportementales)
- motivation, objectifs, etc.

Adaptation contextuelle

Modification de l'information présentée ou du comportement de l'appareil selon le contexte





Bluetooth

Appareils disponibles dépendant de la distance ou de l'emplacement de l'utilisateur

Boutons de volume

Comportement dépendant de l'application à l'avant plan

Multimédia vs Notifications vs Alarmes

Adaptation contextuelle

Modification de l'information présentée ou du comportement de l'appareil selon le contexte

Google Map

- Mise à jour de l'affichage de la carte selon la location de l'utilisateur
- Présentation de commerces à proximité
- Information relative au trafic routier

Adaptation contextuelle

Déclenchement d'actions à partir du contexte utilisateur

Modèle populaire : ***if this then that*** - si ceci, alors cela

Si [Condition contextuelle simple ou complexe], alors [**Action, adaptation appropriée**]

Adaptation contextuelle

Modèle populaire

- si ceci, alors cela
  if this then that

EXEMPLES

Luminosité écran
Si la luminosité ambiante augmente, **alors** on augmente la luminosité de l'écran.
Si la luminosité ambiante baisse, **alors** diminuer la luminosité de l'écran.

Ne pas déranger
Si CurrentTime est entre X et Y, **alors** active le mode Ne pas déranger

Adaptation contextuelle

Risques

Contexte est difficile à quantifier, estimer, mesurer

Problèmes

Quoi faire lorsque le système fait une mauvaise action à partir d'information contextuelle ?

Adaptation contextuelle

Si [Condition contextuelle simple ou complexe], alors [**Action, adaptation appropriée**]

Il peut être tentant d'ajouter des règles de plus en plus compliquées pour tenter d'éviter les erreurs.

Plus de règles != intelligence
Plus de règles = plus de complexité, plus difficile à comprendre pour le système ET l'utilisateur

Une solution possible « human in the loop »

- L'appareil mesure, détecte et présente l'information
- L'utilisateur interprète et fait les actions

Mais est-ce que c'est une bonne solution dans tous les cas?

Adaptation contextuelle - 4 étapes

Concevoir un système/application avec une adaptation contextuelle

1. Identifier le comportement problématique ou qui pourrait bénéficier d'une adaptation contextuelle
2. Identifier les caractéristiques contextuelles pertinentes à la situation désirée
3. Acquisition du contexte, quels capteurs ou sources d'information pourraient me permettre de quantifier ce contexte
4. Action/adaptation à effectuer en fonction de l'état contextuel

Adaptation contextuelle

Exemple : réveil matin

1. On cherche à réduire les désagréments associés à un réveil matin qui sonne en continu lorsqu'on est déjà réveillé
2. Caractéristique pertinente : état d'éveil
3. Acquisition de l'état d'éveil
    a. Mouvement, si >= seuil pendant X minutes
    b. Location, si changement de pièce
    c. Activité, si activité détectée ET est non-statique
4. On éteint la sonnerie

Activité
Adaptation Contextuelle

5 minutes

Individuellement, décrivez les éléments d'une adaptation contextuelle **déjà mis en œuvre sur votre appareil mobile**

1. Problématique qui peut être résolue par une adaptation contextuelle
2. Caractéristiques contextuelles
3. Acquisition du contexte
4. Action/adaptation

Activité
Adaptation Contextuelle

5 minutes

Individuellement, décrivez les éléments d'une adaptation contextuelle **que vous aimeriez que votre téléphone intègre**

1. Problématique qui peut être résolue par une adaptation contextuelle
2. Caractéristiques contextuelles
3. Acquisition du contexte
4. Action/adaptation

Informatique mobile

Portabilité

Suit l'utilisateur dans sa vie de tous les jours

Connectivité

Technologies de télécommunications et de connexion sociale

Sensibilité contextuelle

Adapté aux contextes changeants de l'utilisateur

Informatique ubiquitaire

Principes généraux

Selon la vision originale de **Mark Weiser** (1991)

- La raison d'être d'un ordinateur est de nous aider à faire quelque chose
- Le meilleur ordinateur est un serviteur **invisible** et **silencieux**
- L'ordinateur devrait être une **extension de notre conscience**
- La technologie devrait créer le calme

Informatique mobile	Informatique ubiquitaire
Quelques appareils sont utilisés et transportés	Un très grand nombre d'ordinateurs sont distribués dans l'environnement
Utilisation explicite consciente, à avant-plan	Utilisation implicite, inconsciente, arrière-plan
Compréhension contextuelle **partielle**	Compréhension contextuelle **absolue**
Écosystèmes fermés	Interconnexion complète

Mobile vs Ubiquitaire

Pervasif

Ordinateurs, capteurs et actionneurs distribués dans l'environnement (en incluant, sur les utilisatrices et utilisateurs)

Arrière-plan

L'utilisatrice ou l'utilisateur n'interagit pas de manière consciente avec un ordinateur, elle ou il ne fait que vaquer à ses occupations en interagissant avec son environnement de manière naturelle

Connectivité

- Les ordinateurs distribués communiquent entre eux

- Réseau d'appareils hétérogènes

- Multiples systèmes et protocoles de communications

- Capacité de calcul est distribuée ou centralisée

Contexte

- Les mesures, effectuées par l'ensemble des capteurs et appareils sont combinées afin d'obtenir une compréhension absolue du contexte de l'utilisatrice ou de l'utilisateur.

- Compréhension contextuelle absolue: Le système sait absolument **toute** information qui pourrait lui être nécessaire pour soutenir l'utilisatrice ou l'utilisateur, ex., ses émotions, ses intentions, ses objectifs, l'historique de ses actions, sa perception de X, ses intérêts pour XYZ, etc.

Interactivité

- L'environnement répond aux actions de l'utilisatrice ou de l'utilisateur de manière appropriée selon son contexte.

- L'environnement initie des actions de manière appropriée selon le contexte de l'utilisatrice ou de l'utilisateur.

Caractéristiques

Informatique ubiquitaire

Exemple

Imaginez *que l'UQAC ait une infrastructure d'informatique ubiquitaire* ***idéale et complète***

- **Je quitte mon bureau pour venir au local de la classe**
- **En quittant mon bureau**
  - L'ensemble de mes appareils tombent en mode « ne pas déranger »
  - Les lumières s'éteignent
  - L'ascenseur est appelé de manière à ce qu'il soit là quand j'arrive
- **L'ascenseur m'apporte à l'étage désiré**
- **En sortant de l'ascenseur**
  - Le projecteur s'allume automatiquement, l'écran descend
  - Les lumières se tamisent
  - Il affiche une vue d'ensemble du cours que j'avais préparé
- **Pendant le cours**
  - Les diapositives changent automatiquement, suivant la présentation
  - J'ai fait une petite parenthèse, une nouvelle diapositive est créée à partir des points que j'ai soulevés
  - Un étudiant pose une question. Il montre un croquis qu'il a fait sur sa tablette pour illustrer la question. Le croquis transféré et montré sur le projecteur afin que l'ensemble du groupe puisse le voir
- **Le cours est terminé**
  - Les diapositives sont mises en ligne automatiquement pour que vous y ayez accès
  - Le projecteur s'éteint, l'écran se relève
  - Les lumières s'allument, puis s'éteignent automatiquement lorsque la dernière personne quitte la salle

Activité
Informatique Ubiquitaire

5 minutes

Individuellement, réfléchissez à un aspect de votre vie, une activité que vous faites ou un milieu dans lequel vous passez beaucoup de temps.

Imaginez et écrivez, comme on vient de voir, comment les choses fonctionneraient dans un contexte où un système d'informatique ubiquitaire **idéal** serait en place.

Contrôle

- Qui est en contrôle du comportement du système ?

- L'utilisatrice ou l'utilisateur dispose-t-elle/il d'assez d'information pour comprendre ce qui se passe ?

- Dans quelle mesure l'utilisatrice ou l'utilisateur peut-elle/il exercer un contrôle sur les adaptations contextuelles mises en place ?

Confidentialité et sécurité

- Collecte de données massive sur l'ensemble des facettes de la vie des utilisateurs

- Risque de brèches, dissémination d'information personnelle identifiable

- Perte de contrôle du système

Équité et accessibilité

Assurer l'accès des technologies de manière à ce que l'ensemble de la population puisse profiter de ses avantages

Responsabilité

Les entreprises offrant des produits d'informatique ubiquitaire doivent être tenues responsables de l'entreposage et utilisation des données collectées

Transparence

- Quelles données sont collectées et pourquoi ?

- Quel est l'état du système ? Sa compréhension du contexte ?

Éthique (informatique ubiquitaire)

Informatique persuasive

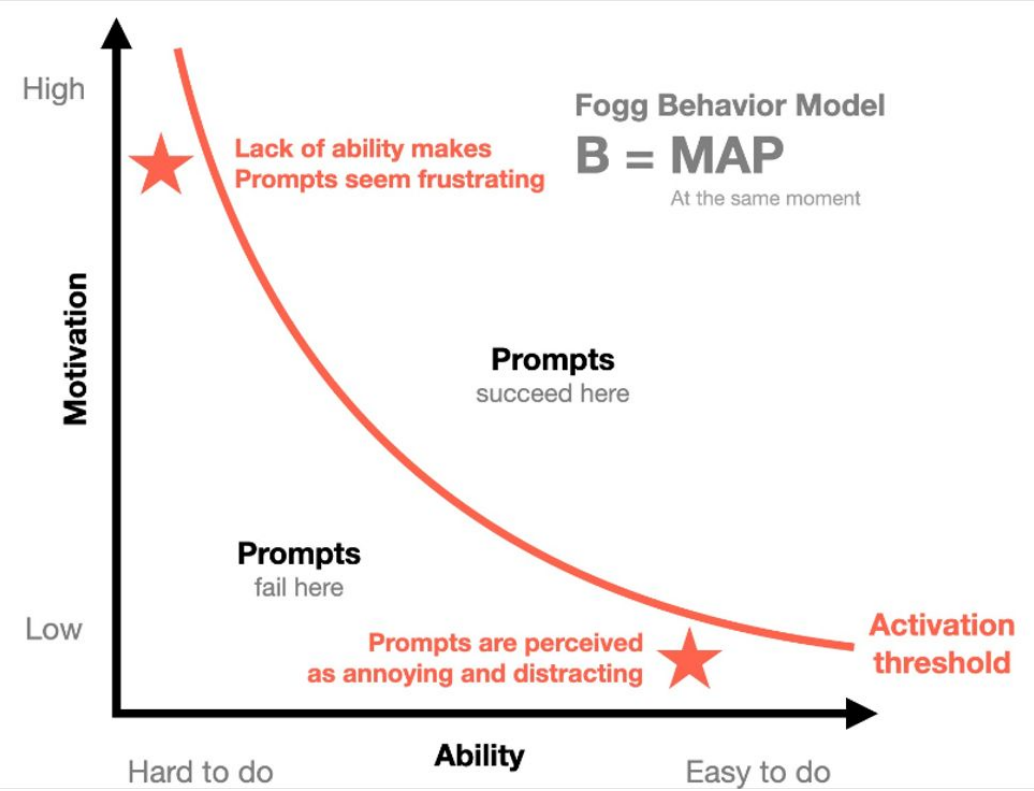
Qu’est ce que c’est ?

Branche de l’informatique qui s'intéresse au potentiel de persuasion des technologies

Spécifiquement, comment motiver, encourager, convaincre des utilisateurs à adopter, développer et maintenir des comportements **positifs** , cohérents avec leurs motivations et objectifs

https://ui-patterns.com/blog/making-the-fogg-behavior-model-actionable

Modèle comportemental de Fogg



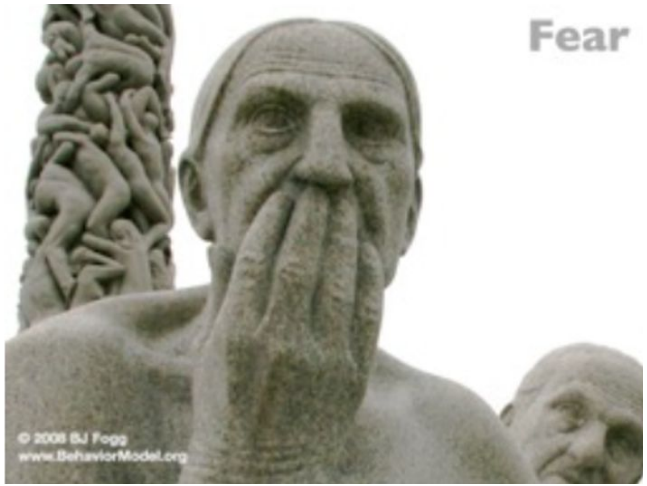
https://ui-patterns.com/blog/making-the-fogg-behavior-model-actionable

Modèle comportemental de Fogg

Motivation de Fogg. **Sensation**

https://behaviormodel.org/motivation

Motivation de Fogg. **Anticipation**

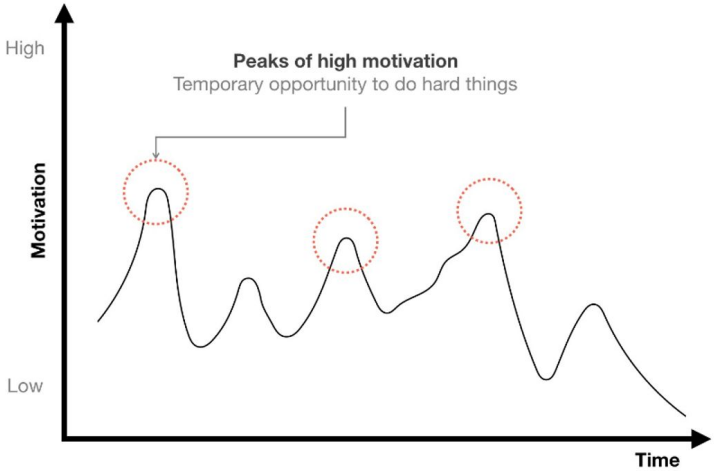


https://behaviormodel.org/motivation

Motivation de Fogg. **Sociale**

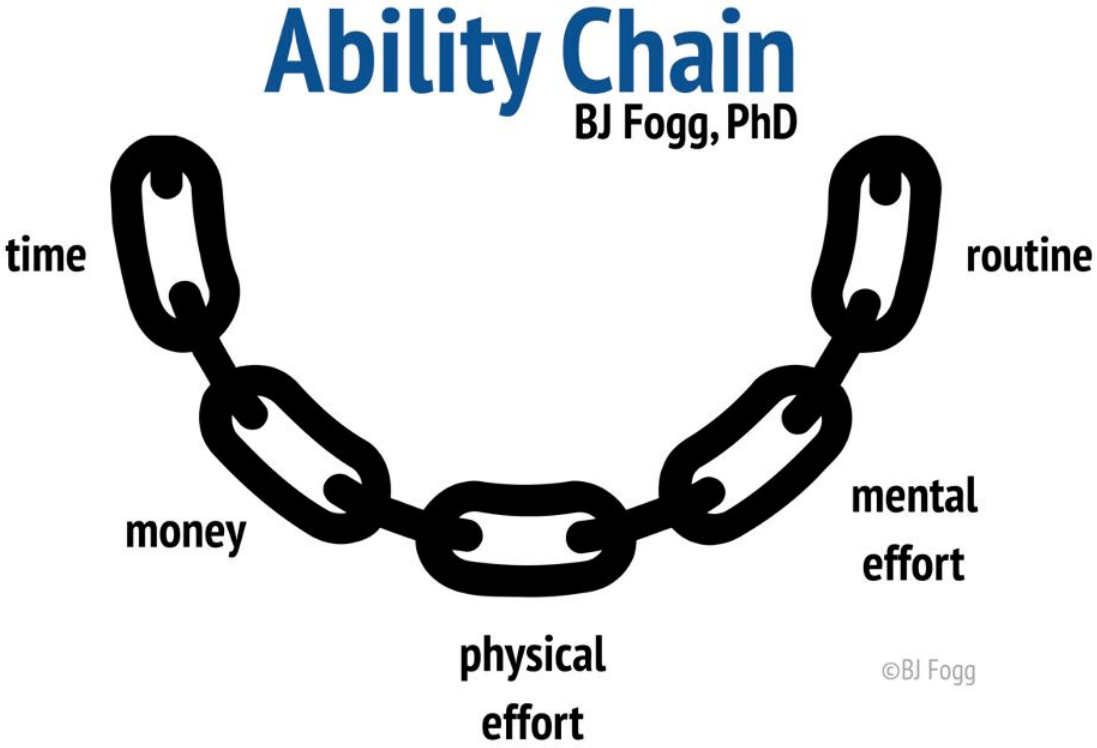
https://behaviormodel.org/motivation

Vagues de motivation



« **Aider les utilisateurs** à adopter les **comportements les plus désirables** qui sont en adéquation avec leur **motivation actuelle**. »

Fogg

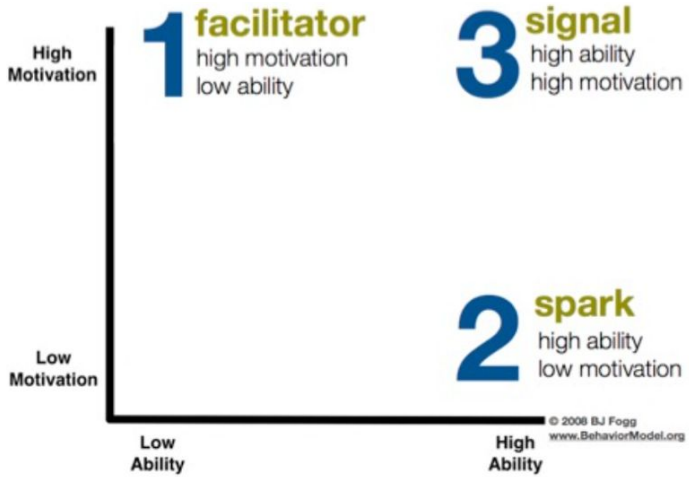


https://behaviormodel.org/ability

Habileté/capacité (de Fogg)

Déclencheurs (de Fogg)

1. Doit être perceptible
2. Doit être associé à un comportement
3. Doit être envoyé à un moment où la motivation et les aptitudes de l'utilisateur sont alignées
4. Doit être envoyé à un moment qui minimise les interruptions et perturbations des activités de l'utilisatrice ou utilisateur.



Déclencheurs - Facilitateurs

Utiliser lorsque la motivation est **élevée** et l'habileté **faible**

Objectif

Guider l'utilisateur dans la réalisation de comportements structurants qui viendront réduire les efforts nécessaire pour réaliser les comportements dans le future

Par exemple dans un contexte d'exercices physiques

- *S'abonner au gym et planifier des séances hebdomadaires avec un entraîneur à domicile. Comme l'entraîneur arrive directement chez toi, les efforts sont minimaux VS se convaincre de se déplacer d'aller au gym. L'évitement du comportement nécessitera aussi des efforts tel que appeler, annuler, re-planifier, etc.*
- *Apprendre de nouveaux types de mouvements. Cela permet d'améliorer l'habileté et la compétence de l'utilisateur*

Déclencheurs - Facilitateurs

Utiliser lorsque la motivation est **faible** et l'habileté **élevée**

Objectif

Mettre l'accent sur un/des **motivateurs** pour encourager la réalisation de petits comportements

Par exemple dans un contexte d'exercices physiques

- *Sensation – rappeler l'inconfort/douleur que la personne ressent après avoir passé plusieurs heures assise*
- *Anticipation – rappeler l'excitation associé à un évènement pour lequel on se prépare*
- *Sociale – mentionner qu'un(e) ami(e) a presque atteint ses objectifs d'activité physique et qu'une petite marche jusqu'à la fontaine suffirait à atteindre les vôtres*

Déclencheurs - Facilitateurs

Utiliser lorsque la motivation est **élevée** et l'habileté **élevée**

Objectif

Rappeler à l'utilisatrice ou à l'utilisateur qu'elle ou il peut réaliser un comportement à un moment donné

Par exemple dans un contexte d'exercices physiques

- *Suggérer de se stationner un peu plus loin de la porte d'entrée du bâtiment*
- *Suggérer d'utiliser les escaliers plutôt que l'ascenseur*
- *Suggérer de profiter d'une pause au travail pour aller chercher de l'eau à la fontaine un peu plus éloignée*

Conseils de conception de systèmes persuasifs

Méthode de Fogg

1. Réalisez et prenez avantage de la motivation à un moment donné, peu importe son niveau. Évitez de tenter d'augmenter la motivation de manière artificielle
2. Guidez les utilisateurs vers des comportements structurants
3. Se concentrer sur la réalisation de petites actions pour avoir un effet durable. De trop grands « pas » échouent presque toujours
4. De petites actions et habitudes grandissent naturellement. Le succès entraîne le succès

Contrôle

- L'utilisatrice ou l'utilisateur dispose-t-elle/il d'assez d'information pour comprendre ce qui se passe ?

- Dans quelle mesure l'utilisatrice ou l'utilisateur peut-elle/il exercer un contrôle sur les objectifs que le système tente d'atteindre ?

Confidentialité et sécurité

- Collecte de données massive sur l'ensemble des facettes de la vie des utilisateurs

- Perte de contrôle du système

Équité et accessibilité

Assurer l'accès des technologies de manière à ce que l'ensemble de la population puisse profiter de ses avantages

Responsabilité

Les entreprises offrant des produits d'informatique persuasive doivent être tenues responsables de l'impact (ou potentiel impact) qu'elles ont sur la vie des utilisateurs

Transparence

- Quelles données sont collectées et pourquoi ?

- Quels sont les objectifs du logiciel ?

Éthique (informatique persuasive)

Activité
Flux d’utilisation

Activité
(20 minutes)

Flux d’utilisation

Votre tâche

1. **Individuellement** (~ 10 minutes)
    a. Dessinez à la main un diagramme de flux d’utilisation représentant les parties pertinentes en lien avec le concept de votre application de projet

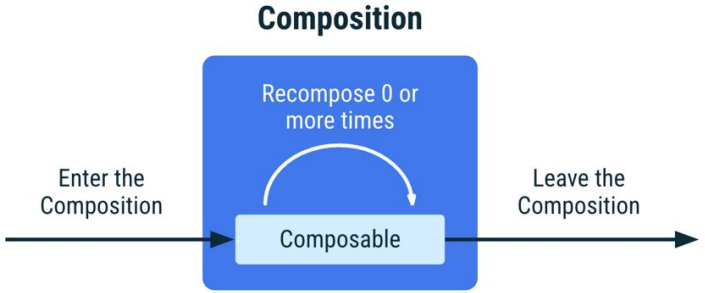
2. **En équipe** (~ 10 minutes)
    a. Présentez vos flux d’utilisation
    b. Identifiez les éléments que vos flux d’utilisation ont en commun et ceux qui sont différents
    c. Éliminez, combinez ou modifiez de manière à générer soit un seul flux d’utilisation complet soit plusieurs flux d’utilisation, mais pour des parties distinctes
    d. Identifiez le chemin idéal d’une couleur (ou symbolique) différente

Interactions et états
(avec Jetpack Compose)

(re)Composition

Les fonctions Compose décrivent l'interface (UI), leur exécution est appelée ***Composition***

- Composition initiale : création d'une composition en exécutant les composables pour la première fois
- **Recomposition** : réexécutez (autant de fois que nécessaire) des composables pour mettre à jour la composition lorsque les données (l'état) changent



L'état

L'**état** détermine/décrit ce qui s'affiche dans l'interface (UI) à un **moment donné**

- Messages les plus récents reçus dans une messagerie instantanée
- Photo de profil de la personne utilisatrice
- Position de défilement dans une liste d'éléments

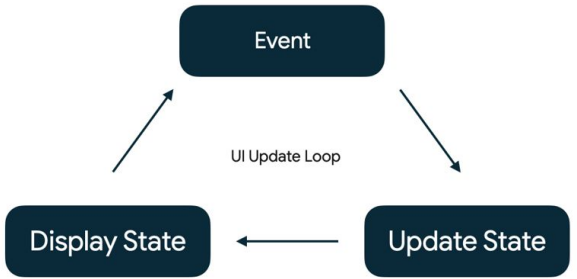
  …

Les événements

L'état est. Les événements surviennent.

Les événements constituent le mécanisme qui permet de modifier l'état, entraînant des changements dans l'interface (UI)

- Un événement est généré par la personne utilisatrice ou une autre partie du programme (lecture d'un capteur, retour d'un service, etc.)
- Un gestionnaire d'événements modifie l'état utilisé par l'interface
- L'interface est actualisée pour afficher le nouvel état



Suivi (observation) d'état

```
@Composable
fun WaterCounter(modifier: Modifier = Modifier) {
    Column(modifier = modifier.padding(16.dp)) {
        // Changes to count are now tracked by Compose
        val count: MutableState<Int> = mutableStateOf(0)

        Text("You've had ${count.value} glasses.")
        Button(
            onClick = { count.value++ },
            Modifier.padding(top = 8.dp)
        ) {
            Text("Add one")
        }
    }
}
```

Utilisation des types `State` et `MutableState<T>`, et de la fonction `mutableStateOf`

(ou d'autres variantes, ex., `mutableIntStateOf`, `mutableFloatStateOf` ...)

Mémorisation d'état

```
@Composable
fun WaterCounter(modifier: Modifier = Modifier) {
    Column(modifier = modifier.padding(16.dp)) {
        // val count: MutableState<Int> = remember { mutableStateOf(0) }
        // Text("You've had ${count.value} glasses.")
        var count by remember { mutableStateOf(0) }
        Text("You've had $count glasses.")
        Button(
            onClick = { count++ },
            Modifier.padding(top = 8.dp)
        ) {
            Text("Add one")
        }
    }
}
```

Utilisation de la fonction `remember` ou `rememberSaveable`

(et éventuellement du mot clé `by` comme délégué de la propriété)

Hisser un état

(state hoisting)

Faire remonter un état vers l'appelant d'un Composable, pour rendre ce dernier sans état *(stateless)*, par exemple :

- pour améliorer sa réutilisabilité
- pour partager l'état avec un autre Composable

```
@Composable
fun StatelessCounter(
    count: Int,
    onIncrement: () -> Unit,
    modifier: Modifier = Modifier
) {
    Column(modifier = modifier.padding(16.dp)) {
        Text("You've had $count glasses.")
        Button(
            onClick = onIncrement,
            Modifier.padding(top = 8.dp)
        ) {
            Text("Add one")
        }
    }
}

...

@Composable
fun Blablabla(modifier: Modifier = Modifier) {
    var count by rememberSaveable { mutableStateOf(0) }
    StatelessCounter(count, { count++ }, modifier)
}
```

Activité
Croquis

Activité (1/2)
(5 minutes)

Croquis

Sortir crayon/stylo et papier

(si nécessaire, des feuilles sont présentes sur mon bureau)

Échauffement

- 5 lignes droites horizontales
- 5 lignes droites verticales
- 5 lignes droites diagonales
- 5 carrés de différentes tailles
- 5 triangles de différentes tailles
- 5 cercles de différentes tailles
- 10 faux mots (gribouillis)

Activité (2/2)
(25 minutes)

Croquis

Votre tâche

1. **En équipe** (~ 5 minutes)
    a. Choisissez deux parties (ex., un menu, une liste, un bouton, un écran particulier) du concept de votre application
2. **Individuellement** (~ 10 minutes)
    a. Pour chaque partie choisie, dessinez à la main au moins 3 croquis représentant différentes manières de les interpréter et d'interagir avec
3. **En équipe** (~ 10 minutes)
    a. Présentez vos croquis
    b. Combinez ou modifiez vos croquis de façon à retenir la meilleure manière pour vous d'implémenter les parties de votre concept

Liste et style (Material Design)

En autonomie (exercices notés)
À commencer à la fin de la séance

Exercices
(tutoriels officiels)

Comprendre les listes et le style avec Material Design

1. Créez une branche à partir de votre branche `Main` et nommez-la `Affirmation`
2. Effectuer l'exercice : https://developer.android.com/codelabs/basic-android-kotlin-compose-training-add-scrollable-list?hl=fr#0
    - Commit/push après les étapes 1, 2 et 3
3. Créez une autre branche à partir de votre branche `Main` et nommez-la `Woof`
4. Effectuer l'exercice : https://developer.android.com/codelabs/basic-android-kotlin-compose-material-theming?hl=fr#0
    - Commit/push après les étapes 3, 4, 5, 6, 7

ATTENTION, il y a de très légères erreurs typographiques dans les énoncés. Vous devez être en mesure de les comprendre et les corriger.

Pour la prochaine séance

- **Exercices** (tutoriels officiels) **à finir**
- **Page de présentation des équipes** (sur le Wiki de votre dépôt du projet - critères sur Moodle)

(Rappel, présentation du concept de votre projet dans 2 semaines, critères sur Moodle)

Lectures et vidéos pertinentes

Informatique Ubiquitaire

- Mark Weiser. 1999. The computer for the 21st century. SIGMOBILE Mob. Comput. Commun. Rev. 3, 3 (July 1999), 3–11. https://doi.org/10.1145/329124.329126
- Leah Buechley - The Democratization of Ubiquitous Computing https://www.youtube.com/watch?v=VrgF0mRGvtk
- Bill Buxton - Ubiquitous Computing and the Emerging Digital Eco-System https://www.youtube.com/watch?v=bSLhQA2OQB4

Informatique Persuasive

- BJ Fogg. 2009. A behavior model for persuasive design. In Proceedings of the 4th International Conference on Persuasive Technology (Persuasive '09). Association for Computing Machinery, New York, NY, USA, Article 40, 1–7. https://doi.org/10.1145/1541948.1541999
- BJ Fogg - Why Tiny Habits Give Big Results https://www.youtube.com/watch?v=g56aKi-z05w

Idées d'activités d'enrichissement personnel

- Lire les articles et regarder les vidéos suggérées *(diapositivc prćcćdcntc)* ou d'autres pertinentes aux sujets abordés
- Rédiger une feuille récapitulative capturant les éléments importants de l'informatique mobile, ubiquitaire et persuasive, compléter l'information en faisant des recherches personnelles
- Réfléchir à la conception d'une application visant à modifier les comportements d'utilisateurs, explorer comment les concepts d'habileté, motivations et déclencheurs pourraient être utilisés