



Lesson 28

21.03.2024

```
public class Test1 {  
    public static void main(String[] args) {  
        byte a = 12;  
        byte b = 13;  
        byte result = a + b;  
        System.out.println(result);  
    }  
}
```



```
public class Test2 {  
    public static void main(String[] args) {  
        int x = 011;  
        int y = 0xfee;  
        int result = x + y;  
        System.out.println(result);  
    }  
}
```

```
public class Test3 {  
    public static void main(String[] args) {  
        int i = 0;  
        int j = 9;  
        do {  
            i++;  
            if (j-- < i++) {  
                break;  
            }  
        } while (i < 5);  
        System.out.println(Integer.toString(i)  
            + Integer.toString(j));  
    }  
}
```

```
public class Test4 {  
    public static void main(String[] args) {  
        int[] x = new int[3];  
        System.out.println("x[0] is " + x[0]);  
    }  
}
```

```
public class Test5 {  
    public static void main(String[] args) {  
        int x = 5;  
        int y = 6;  
        if ((y = 1) == x)  
            System.out.println(y);  
        else  
            System.out.println(++y);  
    }  
}
```

```
public class Test6 {  
    public static void main(String[] args) {  
        for (int i = 0; i < 3; ) {  
            System.out.println("Java");  
        }  
    }  
}
```



5 java questions



1. Модифікатори доступу
2. Що таке цикли в Java
3. Тернарний оператор
4. brake / continue
5. Принципи ООП

Інтернаціоналізація


Питання інтернаціоналізації інтерфейсу користувача - одне з важливих питань при розробці програми. Для цього недостатньо використовувати Unicode і перекласти на потрібну мову всі повідомлення інтерфейсу користувача. Інтернаціоналізація програми означає щось більше, ніж підтримка Unicode. Дата, час, грошові суми і навіть числа можуть по-різному представлятися різними мовами.

Широкого поширення набули умовні скорочення термінів інтернаціоналізації та локалізації додатків **i18n** та **l10n**, у яких цифра означає кількість символів між першою та останньою позицією:


i18n - інтернаціоналізація (internationalization);

l10n – локалізація (localization).

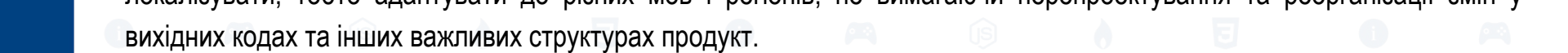




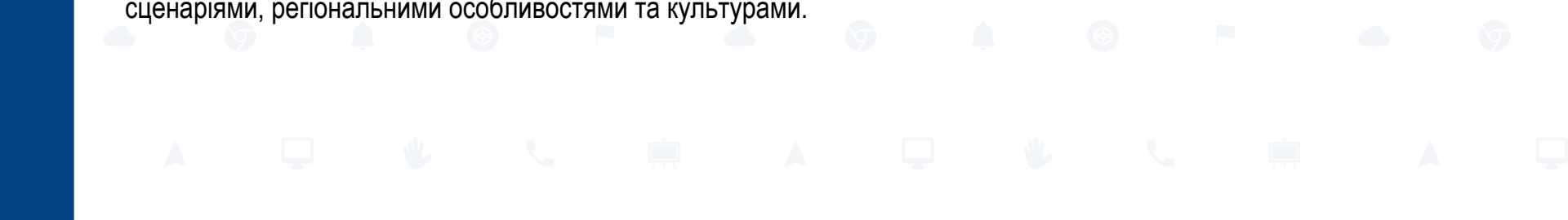
Локалізація (l10n): процес адаптації послуги, продукту чи вмісту до певної місцевості чи ринку, насамперед шляхом перекладу всіх необхідних документів і текстів на місцеву мову, а також з використанням місцевих форматів, символів, стилів подання та інших регіональних особливості, які роблять товар конкурентоспроможним на місцевому ринку.



Інтернаціоналізація (i18n): процес проектування та створення продукту, який гарантує, що його можна легко локалізувати, тобто адаптувати до різних мов і регіонів, не вимагаючи перепроєктування та реорганізації змін у вихідних кодах та інших важливих структурах продукт.



Глобалізація (g11n) відноситься до широкого спектру процесів, необхідних для підготовки та запуску продуктів і заходів на міжнародному рівні. Це широкомасштабна концепція, яка стосується багатомовного спілкування та глобальної готовності продуктів і послуг, що дозволяє використовувати продукт у різноманітному середовищі з різними сценаріями, регіональними особливостями та культурами.



Globalisation process

Internationalisation



Localisation



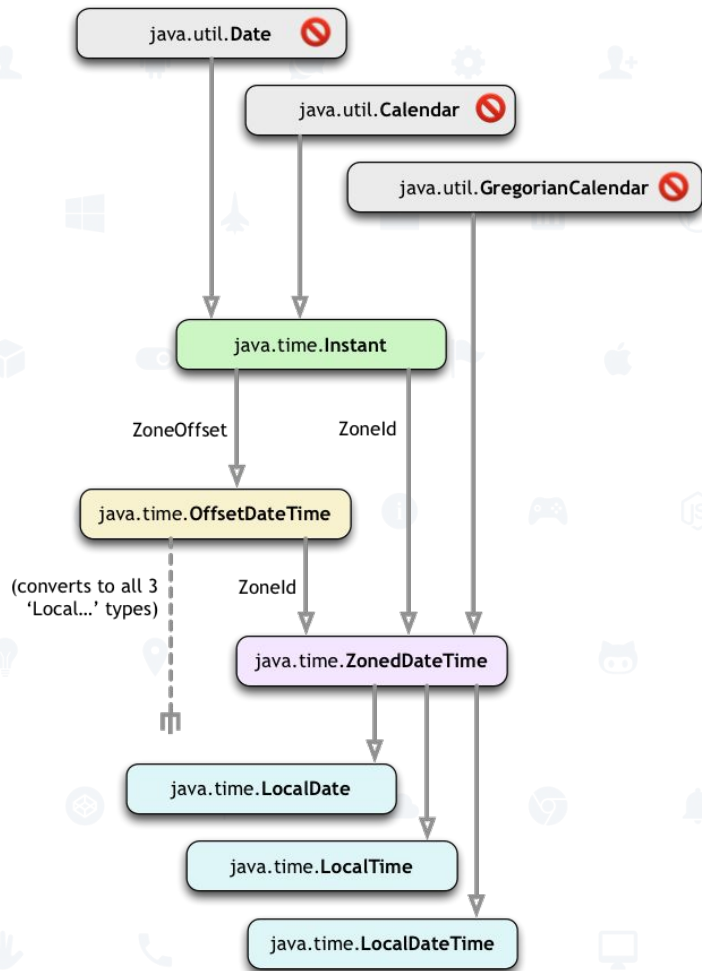
Регіональні стандарти Locale

Додаток, адаптований для міжнародного ринку, легко визначити по можливості вибору мови, для роботи з ним. Але професійно адаптовані програми можуть мати різні регіональні налаштування навіть для тих країн, де використовується однакова мова. У будь-якому випадку команди меню, написи на кнопках та програмні повідомлення мають бути перекладені місцевою мовою, можливо з використанням спеціального національного алфавіту. Але існує ще багато інших більш тонких відмінностей, які стосуються форматів представлення дійсних чисел (розділювачі цілої та дробової частин, роздільників груп тисяч) та грошових сум (включення та місцезнаходження грошового знака), а також формату дати (порядок прямування та символи роздільники днів, місяців та років)

Предопределенные объекты с региональными установками	Объекты, позволяющие указать язык без указания страны
Locale.CHINA	Locale.CHINESE
Locale.FRANCE	Locale.FRENCH
Locale.GERMANY	Locale.GERMAN
Locale.ITALY	Locale.ITALIAN
Locale.JAPAN	Locale.JAPANESE
Locale.US	Locale.ENGLISH

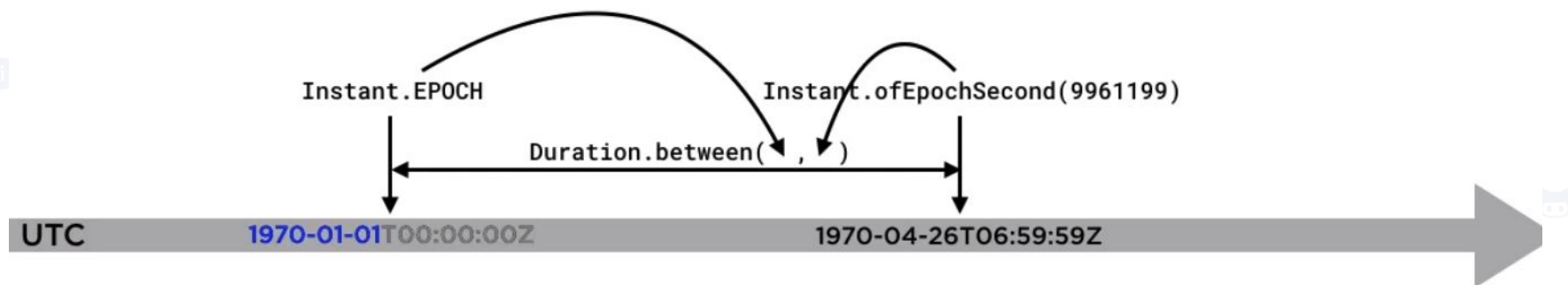
Крім виклику конструктора або вибору обумовлених об'єктів, існує ще два шляхи отримання об'єктів з регіональними налаштуваннями. Статичний метод **getDefault()** класу `Locale` дозволяє визначити регіональне налаштування, яке використовується в операційній системі за замовчуванням. Змінити налаштування за замовчуванням можна викликавши метод **setDefault()**. Однак слід пам'ятати, що даний метод впливає лише Java-програму, а чи не на операційну систему загалом

<i>NumberFormat</i>
<u>+ getInstance() : NumberFormat</u>
<u>+ getCurrencyInstance() : NumberFormat</u>
<u>+ getPercentInstance() : NumberFormat</u>
+ format(in n : double) : String
+ format(in n : long) : String
+ getMaximumFractionDigits() : int
+ getMaximumIntegerDigits() : int
+ setMaximumFractionDigits(in n : int)
+ setMaximumIntegerDigits(in n : int)



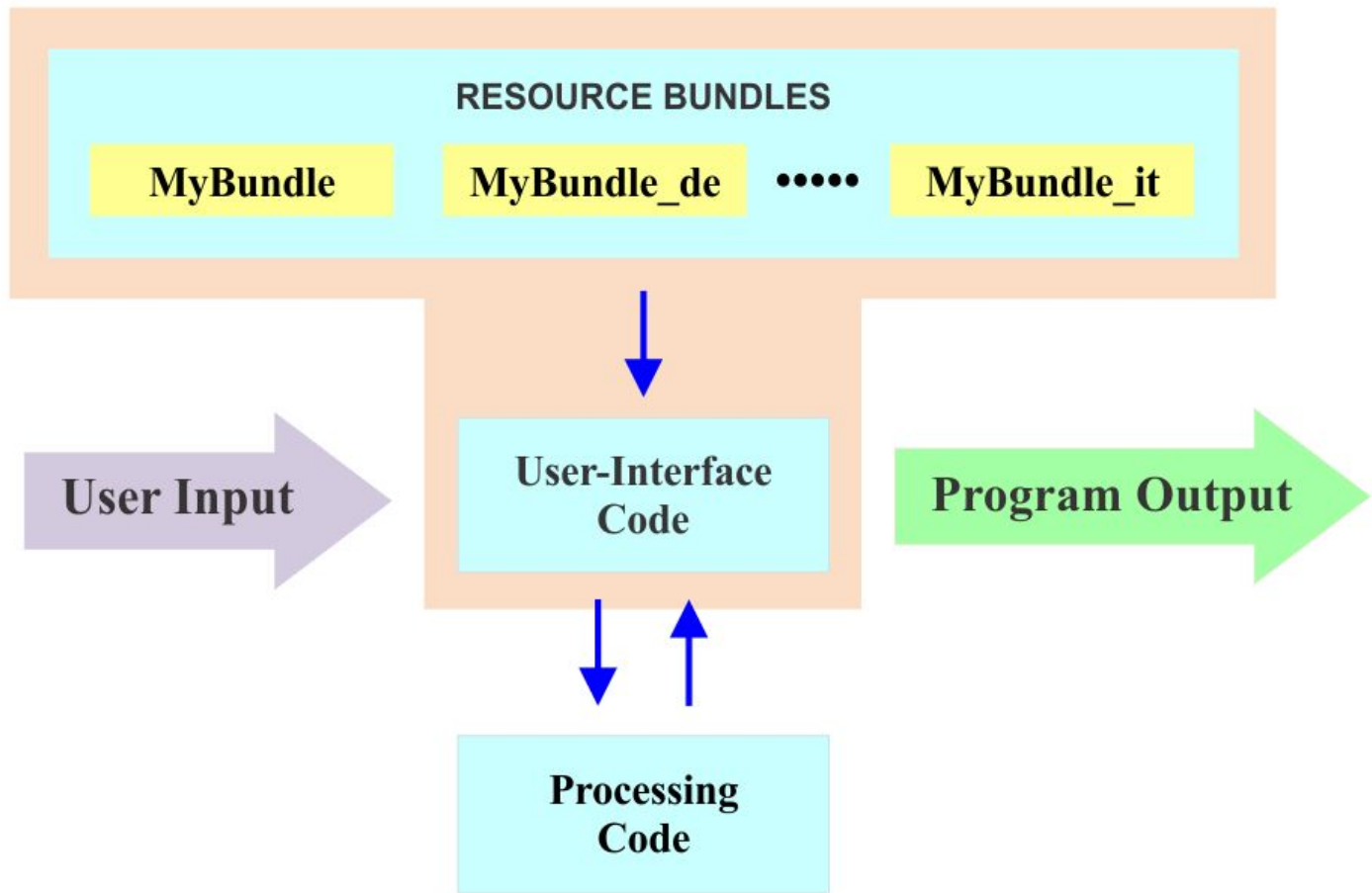
Java Date and Time

Machine/Solar Time





Date-time types in Java & SQL	Legacy class	Modern class	SQL standard data type
Moment in UTC	java.util. Date java.sql. Timestamp	java.time. Instant	TIMESTAMP WITH TIME ZONE
Moment with offset-from-UTC (hours-minutes-seconds)	(lacking)	java.time. OffsetDateTime	TIMESTAMP WITH TIME ZONE
Moment with time zone (`Continent/Region`)	java.util. GregorianCalendar javax.xml.datatype. XMLGregorianCalendar	java.time. ZonedDateTime	TIMESTAMP WITH TIME ZONE
Date & Time-of-day (no offset, no zone) <u>Not</u> a moment	(lacking)	java.time. LocalDateTime	TIMESTAMP WITHOUT TIME ZONE
Date only (no offset, no zone)	java.sql. Date	java.time. LocalDate	DATE
Time-of-day only (no offset, no zone)	java.sql. Time	java.time. LocalTime	TIME WITHOUT TIME ZONE
Time-of-day with offset (impractical, not used)	(lacking)	java.time. OffsetTime	TIME WITH TIME ZONE



Методологія розробки програмного забезпечення (англ. Software development methodology) — сукупність методів, застосовуваних на різних стадіях забезпечення, що мають спільний філософський підхід та, відповідно до цього підходу, дозволяють забезпечити найкращу ефективність процесів розробки.

6 Phases of the Software Development Life Cycle



Agile Modeling

Набір концепцій, принципів і методів (практик), який дозволяє швидко і легко виконувати проектування та документацію для проектів з розробки програмного забезпечення. Не включає в себе докладні інструкції з проектування, містить описи, як побудувати діаграми UML. Основна мета – ефективно моделювання та документація, але не включає в себе програмування і тестування, управління проектом, розгортання та обслуговування системи.

DSDM

Ґрунтується на концепції швидкої розробки додатків (Rapid Application Development, RAD). Це ітеративний і інкрементний підхід, який підкреслює постійну участь користувача/споживача в проекті.

Extreme programming (XP)

Ідея полягає в тому, щоб використовувати корисні традиційні методи розробки по-новому. Наприклад, код що написаний одним девелопером, перевіряється іншим. Також дуже часто використовують парне програмування, де один кодер пише код, а його партнер цей код одразу перевіряє.

Feature driven development (FDD)

Функціонально-орієнтована розробка. Поняття FDD дуже близькі до RUP, єдина різниця: “Кожна функція повинна реалізовуватися не більше двох тижнів.” Якщо завдання досить невелике, його можна розглядати як окрему функцію. Якщо завдання є великим, то він повинен бути розділений на декілька відносно самостійні функції.



Scrum

Встановлює правила для процесу управління розробки програмного забезпечення і дозволяє використовувати існуючу практику кодування, регулювати вимоги або приймати тактичні зміни. Використовуючи цю методику можливо виявити і усунути відхилення від бажаного результату на більш ранніх стадіях розробки програмного забезпечення.

Lean software development

Ця методологія розробки програмного забезпечення метод бережливого виробництва (Lean Manufacturing). Основні принципи цієї методології є: максимальна швидка доставка продукції замовнику з дуже короткими ітераціями, мотивація команди і зосередитися на навчанні з короткими циклами розвитку, раннє тестування і зворотній зв'язок з клієнтами.

Kanban software development

Канбан реалізує принцип «точно в строк» і урівноважує робоче навантаження між усіма членами команди. За допомогою цього методу весь процес розробки зрозумілий для всіх членів команди. Канбан є візуальною моделлю розвитку, яка показує те, що потрібно виробляти, коли і скільки.

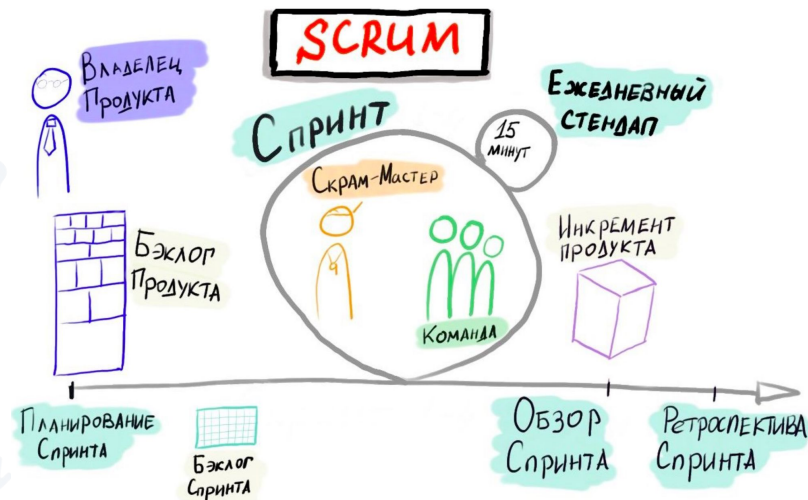
Scrumban

Scrumban є структурою управління і гібрид Scrum і Kanban. Розробники працюють з історіями користувачів і намагаються зберігати ітерації якомога коротшими. Тут не існує конкретних ролей, як наприклад в Scrum. Кожен член команди зберігає свою існуючу роль в проекті.



Методологія "Scrum"

Scrum - гнучкий метод управління проектами. В його основі лежать «спринти» - короткі ітерації, суворо обмежені в часі (зазвичай, 2-4 тижня). Тривалість нарад при цьому зводиться до мінімуму, але збільшується їхня частота. Кожен спринт складається зі списку завдань до кінця ітерації, і кожна операція має свою "вагу". Під час нарад команда обговорює, хто що зробив, що збирається зробити і які проблеми. Для планування у Scrum використовують журнал спринтів. У такому підході в команді часто з'являється Scrum майстер, який налагоджує безперервну роботу всієї команди, створюючи для її комфортні умови. Також на проекті з'являється роль Product Owner. керівника розробки, людини, яка стежить за продуктом та виступає головною ланкою між запитом клієнта та результатом команди





Методологія «Kanban»

Найважливішою особливістю Kanban є візуалізація життєвого циклу проекту. Створюються стовпчики виконання завдань, які здаються індивідуально. Колонки позначені маркерами типу: To do, In progress, Code review, In testing, Done (назва колонок, звичайно ж, може змінюватися). Мета кожного учасника команди - зменшувати кількість завдань у першій колонці. Підхід Kanban наочний і допомагає зрозуміти де виникла проблема. Структуру Kanban не визначають остаточно і безповоротно: залежно від специфіки проекту можна додати імпровізовані колонки. Наприклад, деякі команди використовують систему, у якій потрібно визначити критерії готовності завдання перед виконанням. Тоді додають дві колонки – specify (уточнити параметри) та execute (взятися за роботу)

