Практическая работа №11. Работа с датой и временем

Цель данной практической работы –научиться работать с датами и временем, применять методы класса Date и Calenadar, других классов для обработки строк.

Теоретические сведения

В Java есть много классов, доступных для работы с датой/временем.

Класс Date

Класс Date изначально предоставлял набор функций для работы с датой - для получения текущего года, месяца и т.д. однако сейчас все эти методы не рекомендованы к использованию и практически всю функциональность для этого предоставляет класс Calendar. Класс Date так же определен в пакете java.sql поэтому желательно указывать полностью квалифицированное имя класса Date.

Существует несколько конструкторов класса Date однако рекомендовано к использованию два

Date() и Date(long date) второй конструктор использует в качестве параметра значение типа long который указывает на количество миллисекунд прошедшее с 1 Января 1970, 00:00:00 по Гринвичу. Первый конструктор создает дату использует текущее время и дату (т.е. время выполнения конструктора). Фактически это эквивалентно второму варианту new Date(System.currentTimeMillis); Можно уже после создания экземпляра класса Date использовать метод setTime(long time), для того, что бы задать текущее время.

Для сравнения дат служат методы after(Date date), before(Date date) которые возвращают булевское значение в зависимости от того выполнено условие или нет. Метод compareTo(Date anotherDate) возвращает значение типа int которое равно -1 если дата меньше сравниваемой, 1 если больше и 0 если даты равны. Метод toString() представляет строковое представление даты, однако для форматирования даты в виде строк рекомендуется пользоваться классом SimpleDateFormat определенном в пакте java.text

Классы Calendar и GregorianCalendar

Более развитые средства для работы с датами представляет класс Calendar. Calendar является абстрактным классом. Для различных платформ

реализуются конкретные подклассы календаря. На данный момент существует реализация Грегорианского календаря - GregorianCalendar. Экземпляр этого класса получается вызовом статического метода getInstance(), который возвращает экземпляр класса Gregorian. Подклассы класса Calendar должны интерпретировать объект Date по-разному. В будущем предполагается реализовать так же лунный календарь, используемый в некоторых странах. Calendar обеспечивает набор методов позволяющих манипулировать различными "частями" даты, т.е. получать и устанавливать дни, месяцы, недели и т.д. Если при задании параметров календаря упущены некоторые параметры, то для них будут использованы значения по умолчанию для начала отсчета. т.е.YEAR = 1970, MONTH = JANUARY, DATE = 1 и т.д.

Для считывания, установки манипуляции различных "частей" даты используются методы get(int filed), set(int filed, int value), add(int field, int amount), roll(int field, inr amount), переменная типа int с именем filed указывает на номер поля с которым нужно произвести операцию. Для удобства все эти поля определены в Calendar, как статические константы типа int. Рассмотрим подробнее порядок выполнения перечисленных методов.

Метод set(int field,int value)

Как уже отмечалось ранее данный метод производит установку какого - либо поля даты. На самом деле после вызова этого метода, немедленного пересчета даты не производится. Пересчет даты будет осуществлен только после вызова методов get(), getTime() или TimeInMillis(). Т.о. последовательная установка нескольких полей, не вызовет не нужных вычислений. Помимо этого, появляется еще один интересный эффект. Рассмотрим следующий пример. Предположим, что дата установлена на последний день августа. Необходимо перевести ее на последний день сентября. Если внутреннее представление даты изменялось бы после вызова метода set, то при последовательной установке полей мы получили бы вот такой эффект.

Листинг 11.1 Пример работы с датой и временем

```
public class Test {
public Test() {
}
public static void main(String[] args) {
```

```
    SimpleDateFormat sdf = new SimpleDateFormat("yyyy
MMMM dd HH:mm:ss");
    Calendar cal = Calendar.getInstance();
    cal.set(Calendar.YEAR,2002);
    cal.set(Calendar.MONTH,Calendar.AUGUST);
    cal.set(Calendar.DAY_OF_MONTH,31);
    System.out.println(" Initialy set date:”+
sdf.format(cal.getTime()));
    cal.set(Calendar.MONTH,Calendar.SEPTEMBER);
    System.out.println(" Date with month changed :"+
sdf.format(cal.getTime()));
    cal.set(Calendar.DAY_OF_MONTH,30);
    System.out.println(" Date with day changed:”+
sdf.format(cal.getTime()));

    }}
```

Вывод программы листинга 11.1:

```
Initialy set date: 2002 August 31 22:57:47
Date with month changed: 2002 October 01 22:57:47
Date with day changed: 2002 October 30 22:57:47
```

Рассмотрим еще пример.

Листинг 11.2 – Пример работы с датой и временем

```
import java.text.SimpleDateFormat;
import java.util.Date;
public class DateTest {
public static void main(String[] args) {
Date now = new Date();
System.out.println("toString(): " + now);
// dow mon dd hh:mm:ss zzz yyyy
/* SimpleDateFormat может использоваться для
управления форматом отображения даты/времени:
   E (день недели): 3E or fewer (в текстовом формате
xxx), >3E (в полном текстовом формате)
    M (месяц): M (in number), MM ( в числовом виде,
впереди ноль)
```

```
    3M: (в текстовом формате xxx), >3M: (в полном
текстовом формате)*/
    //   h (часы): h, hh (with leading zero)
    //   m (минуты)
    //   s (секунды)
    //   a (AM/PM)
    //   H (часы 0 до 23)
    //   z (временная зона)
    SimpleDateFormat dateFormatter = new
SimpleDateFormat("E, y-M-d 'at' h:m:s a z");
          System.out.println("Format 1:   " +
dateFormatter.format(now));

          dateFormatter = new SimpleDateFormat("E
yyyy.MM.dd 'at' hh:mm:ss a zzz");
          System.out.println("Format 2:   " +
dateFormatter.format(now));
    dateFormatter = new SimpleDateFormat("EEEE, MMMM
d, yyyy");
          System.out.println("Format 3:   " +
dateFormatter.format(now));
```

Вывод программы листинга 12.2

```
toString(): Sat Sep 25 21:27:01 SGT 2010
Format 1:   Sat, 10-9-25 at 9:27:1 PM SGT
Format 2:   Sat 2010.09.25 at 09:27:01 PM SGT
Format 3:   Saturday, September 25, 2010
```

Класса Date будет достаточно, если вам просто нужна простая отметка времени. Вы можете использовать SimpleDateFormat для управления форматом отображения даты /времени. Используйте класс java.util.Calendar, если вам нужно извлечь год, месяц, день, час, минуту и секунду или манипулировать этими полями (например, 7 дней спустя, 3 недели назад).

Используйте java.text.DateFormat для форматирования даты (от даты до текста) и разбора строки даты (от текста к дате). SimpleDateForamt является подклассом DateFormat.

`Date` является устаревшим классом, который не поддерживает интернационализацию. Calendar и DateFormat поддерживают локализацию (вам нужно учитывать локализацию только в том случае, если ваша программа будет работать во нескольких странах одновременно).

Классы java.util.Calendar и java.util.GregorianCalendar

Рассмотрим пример программы, где мы можем получить год, месяц, день, часы, минуты, секунды:

Листинг 11.3 – Пример использования класса Calendar

```
import java.util.Calendar;
public class GetYMDHMS {
public static void main(String[] args) {
Calendar cal = Calendar.getInstance();
// You cannot use Date class to extract
individual Date fields
int year = cal.get(Calendar.YEAR);
int month = cal.get(Calendar.MONTH);  // 0 to 11
int day = cal.get(Calendar.DAY_OF_MONTH);
int hour = cal.get(Calendar.HOUR_OF_DAY);
int minute = cal.get(Calendar.MINUTE);
int second = cal.get(Calendar.SECOND);
// Заполняем нулями

System.out.printf("Now is %4d/%02d/%02d
%02d:%02d:%02d\n", year, month+1, day, hour, minute,
second);
    }
}
```

Измерение времени

Любые приложения (такие как игры и анимация) требуют хорошего контроля времени. Java предоставляет эти статические методы в классе System. Метод System.currentTimeMillis() возвращает текущее время в

миллисекундах с 1 января 1970 г. 00:00:00 по Гринвичу (известное как «эпоха») в длинном формате.

Измерение прошедшего времени производится как в примере ниже:

```
long startTime = System.currentTimeMillis();
// измерение выполнения кода
  .......
long estimatedTime = System.currentTimeMillis() - startTime;
```

Метод System.nanoTime (): возвращает текущее значение наиболее точного доступного системного таймера, в наносекундах, в течение длительного времени. Введенный с JDK 1.5. метод nanoTime () предназначен для измерения относительного временного интервала вместо предоставления абсолютного времени.

Задания на практическую работу №11

Задание 1. (20%)

Написать программу, выводящую фамилию разработчика, дату и время получения задания, а также дату и время сдачи задания. Для получения последней даты и времени использовать класс Date из пакета java.util.* (Объявление Dated=newDate() или метод System.currentTimeMillis().

Задание 2. (20%)

Приложение, сравнивающее текущую дату и дату, введенную пользователем с текущим системным временем

Задание 3. (20%)

Доработайте класс Student предусмотрите поле для хранения даты рождения, перепишите метод toString() таким образом, чтобы он разработайте метод, возвращал строковое представление даты рождения по вводимому в метод формату даты (например, короткий, средний и полный формат даты).

Задание 4. (10%)

Напишите пользовательский код, который формирует объекты Date и Calendar по следующим данным, вводимым пользователем:

<Год><Месяц><Число>

<Часы1><минуты>

Задание 5 (30%)

Сравнить время выполнения кода в реализации кода в виде различных структур данных из предыдущих заданий (сравнить ArrayList и LinkedList по производительности – операции вставки, удаления, добавления и поиска по образцу)