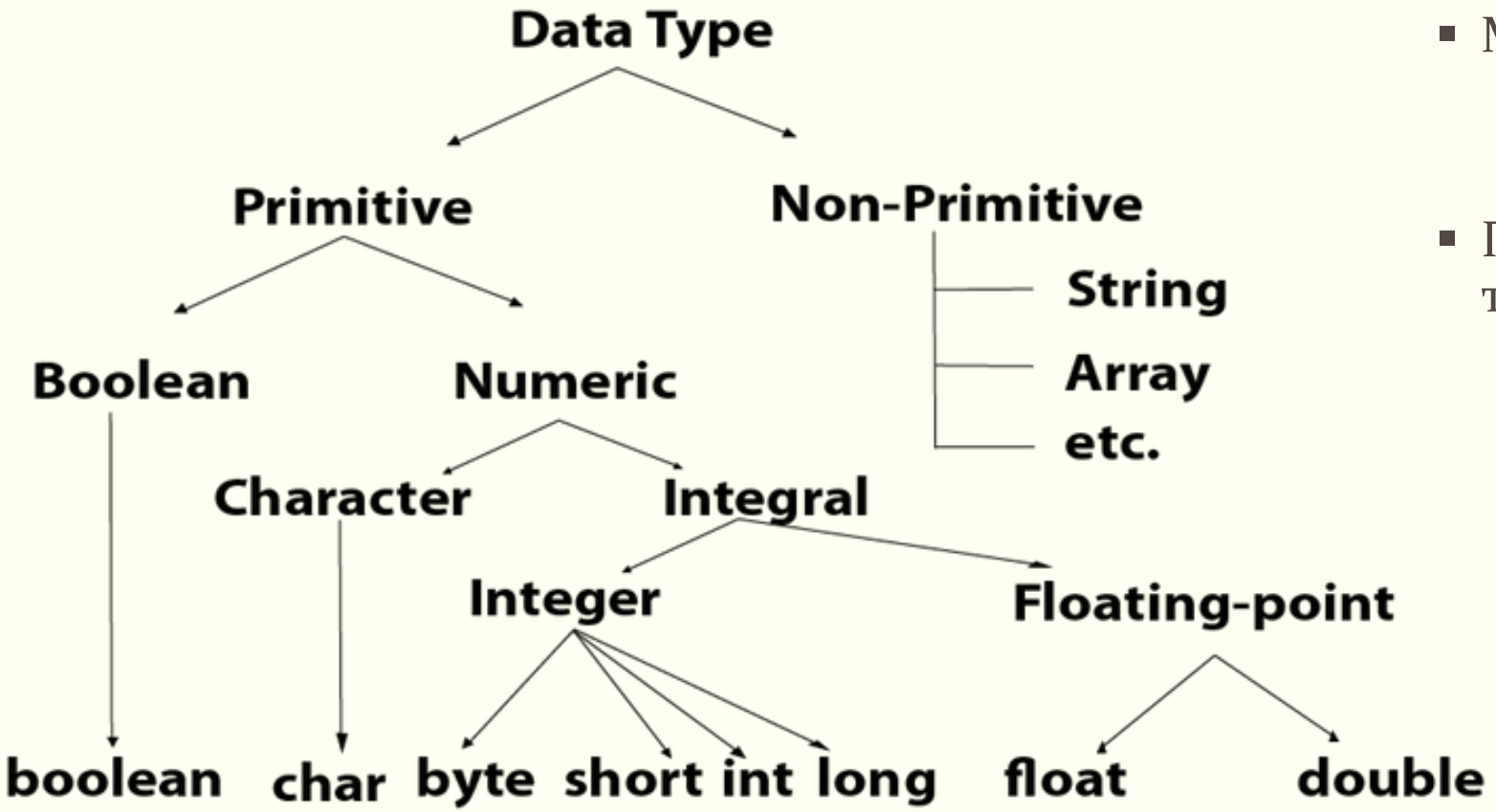
БАЗОВІ ТИПИ ДАНИХ У МОВІ JAVA

Питання 1.2.

Система типів Java



- Мова Java є строго типізованою.
  - Тип кожної змінної має бути оголошено
- Присутні 8 простих (примітивних) типів
  - 4 цілочисельних,
  - 2 з плаваючою комою,
  - 1 з символами в юнікоді,
  - 1 - логічний

Цілочисельні примітивні типи даних

Тип	Требуемый объем памяти (байты)	Диапазон допустимых значений (включительно)
int	4	От **–2147483648** до **2147483647** (т.е. больше **2** млрд.)
short	2	От **–32768** до **32767**
long	8	От **–9223372036854775808** до **–9223372036854775807**
byte	1	От **–128** до **127**

- Зазвичай, найзручніший – int.
- Типи byte і short використовуються в основному в спеціалізованих додатках, наприклад, при низькорівневій обробці файлів або з метою економії пам’яті при формуванні великих масивів.
- Діапазони допустимих даних фіксовані для всіх машин.

Використання суфіксів

- Довгі цілі числа (long) вказуються з суфіксом L (4000000000L).
- Шістнадцяткові числа – з префіксом 0x (0xCAFE), вісімкові – з 0 (010 - не рекомендується для використання, оскільки веде до непорозумінь)
- Починаючи з Java 7, числа можна вказувати у двійковій формі з префіксом 0b, наприклад, 0b1001.
  - Крім того, числові літерали можна тепер вказувати із знаками підкреслення. Наприклад, 1_000_000 для мільйона, що підвищує зручність читання

Числові типи даних з плаваючою крапкою

Тип	Об'єм пам'яті (байт)	Діапазон допустимих значень
float	4	Приблизно ±3.40282347e+38F (6-7 значущих десяткових цифр)
double	8	Приблизно ±1.7976931348623157e+308F (15 значущих десяткових цифр)

- Причинами, за якими все ще застосовують тип float є:
  - Вища швидкість обробки числових даних
  - Економія пам'яті при зберіганні даних (важливо для великих масивів)
- Числові значення типу float вказуються з суфіксом F, наприклад 3.14F.
  - Без такого суфіксу числа відносяться до типу double.
  - Можна використовувати і суфікс D, зокрема 3.14D.
- Числові літерали з плаваючою крапкою можуть представлятись у шістнадцятковій формі.
  - Наприклад, 0.125 = 0x1.0**p**-3, дробова частина записується в шістнадцятковій формі, а показник степені – в десятковій, основа показника степені = 2, а не 10.

Усі операції над числами з плаваючою крапкою відбуваються за стандартом IEEE 754

- Java має три спеціальних дробових значення, які рідко застосовують на практиці:
  - Double.POSITIVE_INFINITY (+∞) – ділення додатного числа на 0
  - Double.NEGATIVE_INFINITY(-∞)
  - Double.NaN (не число) – вираз 0/0, корінь з від'ємного числа тощо
- Аналогічні константи є в типі Float.
  - Усі NaN-величини вважаються різними, проте доступні методи перевірки, наприклад Double.isNaN().
  - if (Double.isNaN(x)) {…}

Тип даних char

- Використовується для представлення окремих символів
  - Не плутайте окремий символ (‘A’) з односимвольним рядком (“A”)
- Кодові одиниці Unicode можуть представлятись шістнадцятковими числами в межах від \u0000 до \uFFFF
  - Символ торгової марки ™ - \u2122
  - Грецька буква π - \u03C0
- Управляючі послідовності символів, що починаються з префікса \u (і ніякі інші) можна навіть вказувати за межами символьних констант або рядків

```
public static void main(String\u005B\u005D args)
```

Управляючі послідовності (escape sequences) спеціальних символів

Управляющая последовательность	Назначение	Значение в уникоде
\b	Возврат на одну позицию	\u0008
\t	Табуляция	\u0009
\n	Переход на новую строку	\u000a
\r	Возврат каретки	\u000d
\"	Двойная кавычка	\u0022
\'	Одинарная кавычка	\u0027
\\	Обратная косая черта	\u005c

- У 1991р. була випущена специфікація Unicode 1.0 (65536 символів), проте в результаті їх не вистачило через занадто великі набори китайської, японської та корейської мов.
  - Нині 16-бітного типу char недостатньо

Вихід із ситуації (в Java SE 5.0+)

- Введемо ряд термінів:
  - ***Кодова точка*** – значення, пов'язане з символом у кодуванні. На Unicode кодові точки записуються в шістнадцятковій формі, якій передує U+ . A = U+0041.
  - В Unicode кодові точки об'єднуються в 17 ***кодових площин***. Перша з них називається основною багатомовною площиною і складається з «класичних» символів Unicode U+0000 - U+FFFF.
  - 16 додаткових площин з кодовими точками від U+10000 до U+10FFFF містять *додаткові символи*.
- Кодування UTF-16 – спосіб представлення в Unicode усіх кодових точок змінної довжини.
  - Символи з багатомовної площини представляються 16-бітовими значеннями, що називають ***кодовими одиницями***.

- Додаткові символи позначаються послідовними парами кодових одиниць.
  - Кожне із значень закодованої таким чином пари потрапляє в область 2048 значень з основної багатомовної площини, що не використовуються.
  - Ця так звана ***область підстановки*** знаходиться в межах від U+D800 до U+DBFF для першої кодової одиниці та від U+DC00 до UDFFF для другої кодової одиниці.
  - Підхід дозволяє відразу визначати, чи значення відповідає конкретному символу, чи є частиною додаткового символу.
  - Алгоритм кодування
- Тип char у Java описує кодову одиницю в кодуванні UTF-16.

Тип boolean

- Тип boolean має два логічні значення: true або false, які слугують для обчислення логічних виразів.
  - Перетворення значень типу Boolean у цілочисельні та навпаки неможливе.

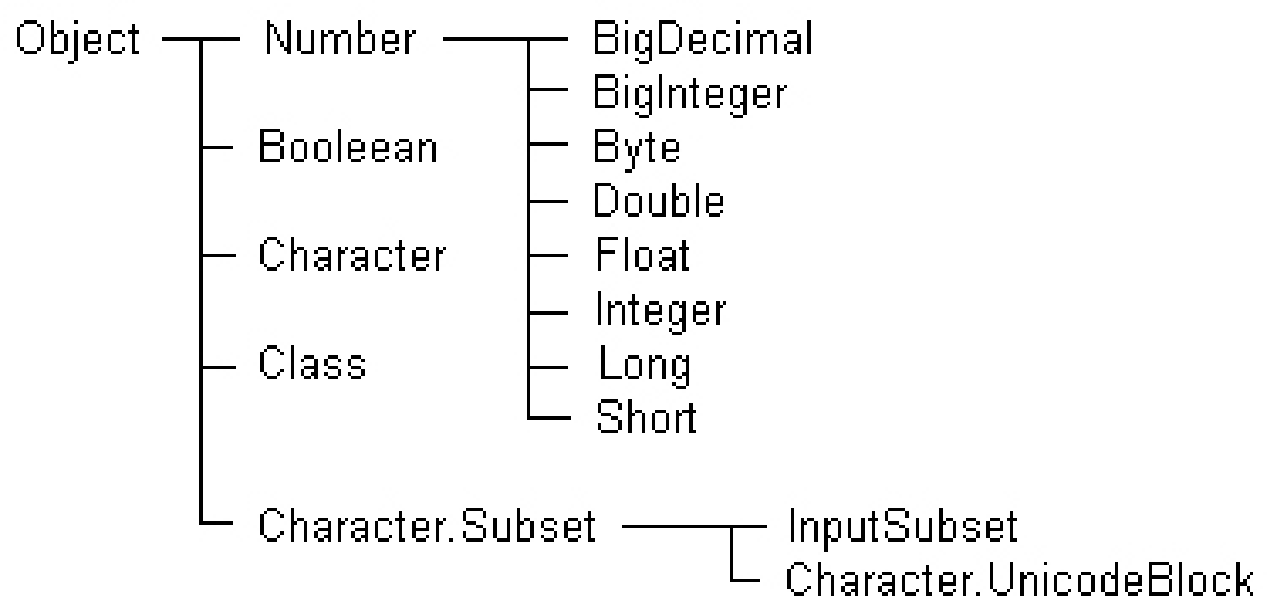
Посилальні типи даних

- Типи даних, для яких у комірці пам'яті (посилкової змінної) містяться не самі дані, а тільки адреси цих даних, тобто посилання на дані.
  - При присвоєнні в посилальну змінну заноситься нова адреса, а не самі дані.
  - Безпосереднього доступу до адреси, що зберігається в посилальних змінних, немає – безпека роботи з даними.
  - Якщо посилальній змінній не присвоєно посилання, в ній зберігається нульова адреса, якій дано символічне ім'я null.
- Посилальні типи Java використовуються в об'єктному програмуванні.
  - Для роботи з рядками, файлами, елементами користувацького інтерфейсу тощо.
  - Всі користувацькі типи, крім типів-перелічень, є посилальними.
    - У тому числі рядкові типи.

Типи-оболонки (wrapper)

- Примітивні типи порушують правило «все в Java є об’єктом», проте вони залишені через багаторічну звичку програмістів.
  - Зручніше працювати зі звичайними числами, ніж з об’єктами класів
- Для примітивних типів є класи-оболонки, які призначені для типових дій при роботі з класами: створення та перетворення об’єктів, отримання значень об’єктів у різних формах та передачі об’єктів у методи за посиланнями.
  - Числові класи мають спільного предка — абстрактний клас Number, у якому описано 6 методів, що повертають відповідне числове значення примітивного типу:
  - byteValue(), doubleValue(), floatValue(), intValue(), longValue(), shortValue().

Ієрархія класів-обгорток



- Усі описані класи, крім Boolean і Class, мають метод compareTo(), який порівнює числове значення, що міститься в даному об’єкті, з числовим значенням об’єкта – аргументом методу compareTo().
- У результаті роботи методу отримуємо ціле значення:
  - 0, якщо значення рівні;
  - від’ємне число (-1), якщо число в даному об’єкті менше, ніж в об’єкті-аргументі;
  - Додатне число (+1), інакше.

Причини створення класів-обгорток

- Робота з колекціями (Collections Framework) включає використання списків, множин (сетів, sets) та відображень (maps), які можуть містити лише об’єкти і не можуть включати значення примітивних типів.
  - Значення примітивного типу зберігається в екземплярі класу-обгортки, який, у свою чергу, міститься в екземплярі колекції.
- Ці класи забезпечують місце для зберігання корисних констант (MAX_VALUE, MIN_VALUE та ін.) та методів класу (наприклад, Integer.parseInt(), Character.isDigit() та ін.).

ДЯКУЮ ЗА УВАГУ!

Наступне запитання: лексичні основи мови програмування Java