

## Зміст (коротко)

Вступ	21
1. Відкидаючи зайве. Швидке запускання	33
2. Подорож до Об'єктивіа. Класи й об'єкти	53
3. Знайте свої змінні. Примітиви та посилання	81
4. Як поводяться об'єкти. Методи використовують змінні екземпляра	103
5. Надпопулярні методи. Створення програм	127
6. Використання бібліотеки Java. Знайоміться: Java API	157
7. Покращене життя в Об'єктивіа. Наслідування та поліморфізм	197
8. Серйозний поліморфізм. Інтерфейси та абстрактні класи	229
9. Життя і смерть об'єктів. Конструктори та збирання сміття	267
10. Числа мають значення. Числа і статичні члени класу	305
11. Небезпечна поведінка. Обробка винятків	347
12. Дуже графічна історія. Створення графічного користувацького інтерфейсу (GUI)	385
13. Попрацюйте над Swing. Робота з бібліотекою Swing	431
14. Збереження об'єктів. Сериалізація та I/O (об'єкт/об'єкт)	461
15. Встановлюємо зв'язок. Мережева та потоки	503
16. Структури даних. Колекції та шаблони	561
17. Випуск коду. Пакети, jar-архіви та розгортання	613
18. Розподілені обчислення. Віддалене розгортання за допомогою RMI	639
<b>Додаток А.</b> Фінальна «Кухня Коду»	681
<b>Додаток Б.</b> Десять важливих речей, що не потрапили до цієї книжки	709
<b>Показчик</b>	709

## Зміст (докладно)

### Вступ

**Ваш мозок і Java.** Коли ви намагаєтеся щось вивчити, ваш мозок намагається надати вам послугу, переконуючи, що все це не варто уваги. Він думає: «Краще перейматися важливішими речами, наприклад, небезпечними дикими тваринами або тим, що катання голяка на сноуборді — погана ідея». Як же переконати свій мозок у тому, що від знання Java залежить ваше життя?

Для кого ця книжка?	22
Кому швидше за все слід триматися якнайдалі від цієї книги?	22
Ми знаємо, про що ви подумали	23
І ми знаємо, чим переймається ваш мозок	23
Ми вважаємо читача «Head First» учнем	24
Металізація: мислення про мислення	25
Ось що зробили МИ	26
Ось що ви можете зробити, аби змусити свій мозок підкорятися	27
Що необхідно для читання цієї книги	28
Деякі останні нюанси, що їх варто знати	29
Технічні редактори	30
Подяки	31

## 1

**Відкидаючи зайве**

**Java відкриває нові МОЖЛИВОСТІ.** Від часів скромного релізу відкритої (і доволі вбогої) версії 1.02, ця мова підкорила розробників своїм дружнім синтаксисом, об'єктно-орієнтованими функціями, управлінням пам'яттю і, що найважливіше, перспективою перенесення на різні платформи. Ми швиденько зануримося, напишемо код, скомпілюємо й запустимо його. Маємо на увазі синтаксис, цикли, розгалуження і все, що робить Java такою кльовою. Отже, починаємо.

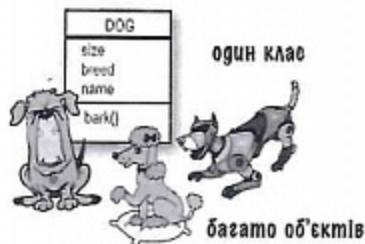


Як працює Java .....	34
Що ви робитимете в Java.....	35
Дуже стисла історія Java.....	36
Структура коду в Java.....	39
Анатомія класу .....	40
Створення класу з методом main .....	41
Що можна сказати всередині main методу? .....	42
Зацикловання, зацикловання і.....	43
Умовне розгалуження.....	45
Створення серйозного бізнес-застосунку.....	46
Понеділковий ранок Боба .....	47
Генератор фраз .....	49
Вправи і головоломки.....	52

## 2

**Подорож до Об'єквілія**

**Мені казали, що там будуть об'єкти.** У Розділі 1 ми розмістили весь свій код у методі main(), але це не зовсім об'єктно-орієнтований підхід. Тепер ми маємо залишити позаду цей світ процедур і почати розробляти власні типи об'єктів. Ми розглянемо, чим цікаве об'єктно-орієнтоване (ОО) програмування за допомогою мови Java. Також розглянемо різницю між класом та об'єктом і дізнаємося, як об'єкти можуть покращити ваше життя.



Війна за крісло .....	60
Що вам подобається в ОО?.....	65
Проектуючи клас, поміркуйте про об'єкти, які буде створено з цього типу класу .....	66
Створюємо перший об'єкт .....	68
Створення і тестування об'єктів Movie .....	69
Швидше! Забирайтеся з головного методу! .....	70
Запускаємо гру «Відгадування» .....	72
Вправи і головоломки.....	74

## 3

## Знайте свої змінні

Змінні бувають двох видів: примітиви і посилання.

У житті має бути щось окрім чисел, рядків і масивів. Як бути з об'єктом PetOwner у якого є змінна типу Dog? Або Car зі змінною Engine? У цьому розділі ми розкриємо завісу таємниці над типами в мові Java й розглянемо, що саме можна оголошувати як змінну, які значення присвоювати їм і що взагалі можна з ними зробити. І нарешті, розглянемо як насправді виглядає життя в кучі (heap), яка контролюється сміттєзбиральним механізмом (garbage-collectible heap).

Тут *heap*, або «куча», — це спеціальна ділянка пам'яті, яка контролюється автоматично і створена для впорядкування об'єктів.



Оголошення змінної.....	82
«Мені, будь ласка, double мокко. Хоча ні, зробіть int» .....	83
Пильнуйте, щоби нічого не розлити.....	84
Тримайтеся подалі від цього ключового слова!.....	85
Керуємо об'єктом Dog .....	86
Посилання на об'єкт — це лише ще одне значення змінної.....	87
Масив як підставка для склянок .....	91
Масиви теж є об'єктами.....	91
Створюємо масив об'єктів Dog.....	92
Контролюйте свій об'єкт Dog (за допомогою посилання).....	93
Приклад класу Dog.....	94
Вправи і головоломки .....	95

## 4

## Як поведуться об'єкти

Стан впливає на поведінку, а поведінка — на стан.

Нам відомо, що об'єкти характеризуються **станом** і **поведінкою**, що представлені **змінними екземплярами** і **методами**. Зараз ми обговоримо зв'язок між станом і поведінкою. Поведінка об'єкта використовує унікальний стан об'єкта. Інакше кажучи, **методи використовують значення змінних екземпляра**. Скажімо, «якщо собака важить менше 14 фунтів, треба видати звук *wurru*, інакше...» **Давайте змінимо стан!**



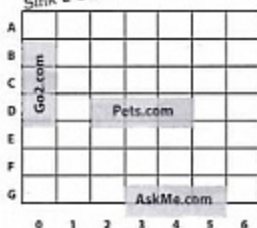
Пам'ятайте: клас описує те, що об'єкт знає і робить .....	104
Розмір впливає на гучність гавкоту.....	105
Ви можете передавати методу різні значення.....	106
Ви можете отримувати значення як результат виконання методу .....	107
Ви можете передати методу одразу декілька значень.....	108
У Java все передається за значенням. Це означає, що значення копіюється під час передавання.....	109
Круті штуки які можна зробити з параметрами і типами повернутих значень ..	111
Інкапсуляція .....	112
Інкапсуляція класу GoodDog.....	114
Як поведуться об'єкти всередині масиву? .....	115
Оголошення та ініціалізація змінних екземпляра .....	116
Різниця між змінними екземпляра та локальними змінними.....	117
Порівнюємо змінні (примітиви або посилання) .....	118
Вправи і головоломки.....	120

## 5

**Надпотужні методи**

Давайте зробимо метод потужнішим. Ви вже побавилися зі змінними, погратися з декількома об'єктами і написали невеличкий код. Але для повноцінної роботи потрібно більше інструментів. На кшталт **операторів** і **циклів**. Це може бути корисним для **генерації випадкових чисел**. І перетворення об'єкта String на Int — так, це було б круто. Чому б вам не створити щось справжнє, аби на власні очі побачити, як з нуля пишуться (і тестуються) програми? **Можливо, це буде гра** на кшталт Sink a Dot Com (подібна до «Морського бою»).

Ми збирасмося створити гру Sink a Dot Com



Давайте створимо аналог «Морського бою» — гру «Sink a Dot Com».....	128
Передусім — високорівневе проєктування .....	129
Плавний вступ у спрощену версію гри .....	130
Розробка класу .....	131
Пишемо реалізацію методу.....	133
Тестовий код для класу SimpleDotCom.....	134
Метод checkYourself() .....	136
Про нові речі.....	137
Псевдокод для класу SimpleDotComGame .....	140
Методи main(), random() та getUserInput() у гри .....	142
Останній клас GameHelper.....	144
Зіграємо! Що це? Помилка?.....	145
Детальніше про цикли for.....	146
Приведення примітивів .....	149
Вправи і головоломки.....	150

## 6

**Використання бібліотеки Java**

Разом із Java постачаються сотні готових класів. Можете не витратити час на винахід власного велосипеда, якщо знаєте, як знайти необхідне в бібліотеці Java (її також називають Java API). Гадаю, у вас знайдуться важливіші справи. При написанні коду зосередьтеся на тій частині, що є унікальною для вашого застосунку. Базова бібліотека Java являє собою гігантський набір класів, готових до застосування як будівельні блоки.

У попередньому розділі ми зупинились на найцікавішому місці. На помилці ...	158
То що ж трапилося? .....	159
Яким чином це виправити? .....	160
Перший варіант занадто незграбний .....	161
Другий варіант трохи кращий, але все ще незграбний .....	161
Прокиньтесь та відчуйте дух бібліотеки.....	164
Декілька прикладів застосування ArrayList .....	165
Порівняння ArrayList зі звичайним масивом .....	168
Виправляємо код класу DotCom .....	170
Новий і вдосконалений клас DotCom .....	171
Давайте створимо справжню гру «Sink a Dot Com» .....	172
Що треба змінити? .....	173
Хто що робить у гри DotComBust (і коли) .....	174
Псевдокод для справжнього класу DotComBust .....	176
Остаточна версія класу DotCom .....	182
Суперпотужні логічні вирази .....	183
Використання бібліотеки (Java API).....	186
Як працювати з API.....	190
Вправи і головоломки.....	193

«Примемо знати, що ArrayList існує тільки в пакеті java.util. Але чим чиним я зможу б додуматися до цього самостійно»

Джуніа, 31 рік, моден



## 7 Покращене життя в Об'єктивлі

**Плануйте свої програми з прицілом на майбутнє.** Чи легко написати код так, щоби хтось інший міг би його розширити? Чи зацікавить вас створення гнучкого коду, що не боїться змін у технічному завданні, що виникають в останню мить? Долучившись до планування поліморфізму, ви дізнаєтеся про п'ять кроків вправного проектування класів, три прийоми задля досягнення поліморфізму і вісім способів створення гнучкого коду, а якщо діятимете енергійно — отримаєте бонусний урок із чотирма порадами щодо використання наслідування.

Війни за крісло переглянуто.....	198
Наслідування в деталях.....	200
Давайте розробимо дерево наслідування для програми-симулятора життя тварин.....	202
Застосування наслідування для запобігання дублюванню коду в підкласах.....	203
Чи всі тварини харчуються однаково?.....	204
Шукаємо нові можливості наслідування.....	205
Який метод викликається?.....	207
Проектування дерева наслідування.....	208
Використання відносин Є (IS-A) і МІСТИТЬ (HAS-A).....	209
Але стривайте! Це ще не все!.....	210
Як дізнатися, що наслідування зроблено правильно?.....	211
Наслідування — добре чи погано?.....	213
У чому ж полягає справжня цінність наслідування?.....	214
Дотримання умов контракту: правила переопиання.....	222
Перевантаження (overloading) методу.....	223
Вправи і головоломки.....	224

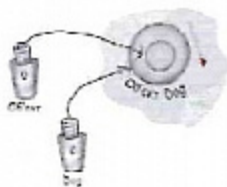


## 8 Серйозний поліморфізм

**Наслідування — це лише початок.** Аби задіяти поліморфізм, необхідні інтерфейси. Нам слід вийти за рамки простого наслідування, щоби ви могли отримати гнучкість лише проєктуючи та кодуючи інтерфейси. Що таке інтерфейс? Це стовідсотково абстрактний клас. Що таке абстрактний клас? Це клас, для якого не можна створити екземпляр. Навіщо це потрібно? Читайте розділ...

Здається ми щось забули, коли розробляли це?.....	230
Компілятор не дозволить вам створити екземпляр для абстрактного класу.....	233
Абстрактний проги Конкретного.....	234
Абстрактні методи.....	235
Ви <b>ПОВИННІ</b> реалізовувати всі абстрактні методи.....	236
Поліморфізм у дії.....	238
Ох-ох! Тепер нам потрібно зберігати ще й об'єкти типу Cat!.....	239
Чому б не зробити клас настільки універсальним, аби він міг приймати будь-що? ...	240
Що ж містить цей ультра-супер-мега-клас Object?.....	241
Використання поліморфних посилань типу Object має свою ціну... ..	243
Коли Dog поводитиметься не як Dog.....	244
Установіть зв'язок зі своїм внутрішнім Object.....	246
Що робити, якщо вам необхідно змінити контракт?.....	250
Розглянемо варіанти проектування для повторного використання існуючих класів у програмі PetShop.....	251
Інтерфейс поспішає на допомогу!.....	256
Створення та реалізація інтерфейсу Pet.....	257
Вправи і головоломки.....	262

```
Object o = a1.get(index);
Dog d = (Dog) o;
d.bark();
```



Приводимо Object до Dog, який, як ми знаємо, мис, мис і с.

## 9

## Життя і смерть об'єктів

**Об'єкти народжуються і вмирають.** Ви головний. Ви вирішуєте, коли і як саме створювати їх. І ви вирішуєте, коли відмовитися від них. **Збирач сміття (Garbage Collector — gc)** звільняє від них пам'ять. Ми розглянемо, як створюються об'єкти, де розміщуються, як їх ефективно використовувати і звільняти пам'ять від них, коли вони стають непотрібними. Тобто ми обговорюватимемо купу (heap), стек (stack), зону впливу (scope), конструктори, суперконструктори, null посилання і вимоги gc.

Коли хтось викликає метод `foo()`, цей об'єкт `Disk` стає недоступним. Його єдине посилання — перепрограмується на інший об'єкт `Disk`.



Змінний «`x`» призначено новий об'єкт `Disk`, при цьому оригінальний (початковий) об'єкт `Disk` стає недоступним. Та перша кінка відтепер засмажена.

Стек і купа: місця для життя.....	268
Як щодо локальних змінних, які є об'єктами?.....	270
Якщо локальні змінні мешкають у стеці, то де перебувають змінні екземпляра? ...	271
Таїнство створення об'єкта.....	272
Створюємо <code>Disk</code> .....	274
Ініціалізація стану нового об'єкта <code>Disk</code> .....	275
Чи завжди компілятор створює конструктор без аргументів? Ні!.....	278
Наноогляд: чотири речі, які слід пам'ятати про конструктори.....	281
Роль конструкторів суперкласу в житті об'єкта.....	283
Як викликати конструктор суперкласу?.....	285
Чи може дитина існувати до появи своїх батьків?.....	286
Конструктори суперкласів з аргументами.....	287
Виклик одного переважаного конструктора з іншого.....	288
Тепер ми знаємо, як народжується об'єкт. Але скільки він живе?.....	290
Як щодо змінних посилань?.....	292
Вправи і головоломки.....	298

## 10

## Числа мають значення

**Займемося математикою.** Java API має методи для роботи з числами на кшталт визначення абсолютного значення, округлення, пошук мінімуму / максимуму тощо. А як щодо форматування? Можливо, ви захочете, щоби числа друкувалися із точністю до двох десяткових знаків або з комами в усіх необхідних місцях. Ви також можете друкувати дати і маніпулювати ними. А як щодо розбору рядка в число? Або перетворення числа на рядок? Ми почнемо цей розділ із вивчення особливостей *статичної змінної* або методу.

Статичні змінні є загальними для всіх екземплярів класу.

Екземпляр дитини № 1      статична змінна: `IceCream`      Екземпляр дитини № 2



Змінні екземпляри по одній на екземпляр

Статичні змінні: по одній на клас

Методи класу <code>Math</code> — найближчі до глобальних.....	306
Різниця між звичайними (не статичними) і статичними методами ( <code>static</code> ) ...	307
Статичні методи.....	308
Статична змінна.....	311
Модифікатор <code>final</code> придатний не лише для статичних змінних... ..	315
Математичні методи.....	318
Обгортка для примітивів.....	319
Автоматичне пакування.....	321
Стривайте! Це ще не все!.....	324
А тепер навпаки... перетворення простого числового значення в <code>String</code> ... ..	325
Форматування чисел.....	326
Про числа ми поговорили. А як щодо дат?.....	333
Переміщення назад і вперед у часі.....	335
Отримання об'єкта, що розширює <code>Calendar</code> .....	336
Робота з об'єктами <code>Calendar</code> .....	337
Ще більше статички! <code>static imports</code> .....	339
Вправи і головоломки.....	342

## 11

## Небезпечна поведінка

**Трапляються різні речі.** То файл зникає. То сервер падає. Хай би яким вправним розробником ви були, неможливо контролювати геть усе. При створенні ризикованого методу вам знадобиться код, що буде обробляти можливі нестандартні ситуації. Але яким чином дізнатися, чи ризиковий цей метод? І куди помістити код для обробки *непередбаченої* ситуації? У цьому розділі ми розробимо музичний MIDI-плеєр, що використовує ризикове JavaSound API.

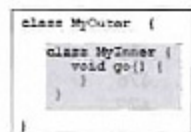


Давайте створимо Музичну Машину.....	348
Розпочнемо з основ.....	349
Для початку нам потрібен Sequencer.....	350
Ризикові методи.....	351
Виняток — це об'єкт... типу Exception.....	354
Керування програмним потоком у блоках try/catch.....	358
Finally: для дій, що їх потрібно виконати НЕЗВАЖАЮЧИ НИ НА ЩО... ..	359
Винятки є поліморфними.....	362
Оминаючи (оголошуючи) виняток, ви лише відкладаєте неминуче.....	368
Повертаємося до нашого музичного коду.....	370
Відтворення звуку.....	372
Створення MIDI-події (даних про композицію).....	375
MIDI-повідомлення: серце MidiEvent.....	376
Змістимо повідомлення.....	377
Версія 2. Використання аргументів командного рядка для експериментів зі звуками... ..	378
Вправи і головоломки.....	380

## 12

## Дуже графічна історія

**Готуйтеся, вам доведеться створювати GUI** (*graphical user interface* — графічний користувацький інтерфейс). Навіть якщо ви впевнені, що все життя будете писати код для серверних програм, рано чи пізно вам доведеться створювати інструменти, і тоді виникне потреба у використанні графічного інтерфейсу. Роботі над GUI присвячені два розділи, із яких ви дізнаєтеся про ключові особливості мови Java, включно з **обробкою подій** і **внутрішніми класами**. Ми розмістимо кнопку на екрані, малюватимемо на екрані, додаватимемо зображення у форматі JPEG і навіть створюватимемо анімацію.



Зовнішні та внутрішні об'єкти відтепер тісно пов'язані.



Ці два об'єкти на картинці мають особливий зв'язок. Внутрішній може використовувати змінні зовнішнього (і навпаки).

Усе починається з вікна.....	386
Отримуємо користувацьку подію.....	389
Повернемося до графіки... ..	395
Створіть власний віджет для малювання.....	396
Що цікавого можна зробити в paintComponent().....	397
За кожним об'єктом Graphics стоїть об'єкт Graphics2D.....	398
Компонування графічних елементів: розміщуємо декілька віджетів у фреймі... ..	402
Спробуємо зробити це із ДВОМА кнопками.....	404
Отже, зараз нам знадобляться ЧОТИРИ віджети.....	404
І нам потрібно отримати ДВІ події.....	404
Внутрішній клас поспішає на допомогу!.....	408
Те, чого ми дійсно прагнемо, являє собою щось на кшталт... ..	415
Відстежуємо подію, не пов'язану з GUI.....	419
Легкий спосіб створення повідомлень/подій.....	420
Приклад: використовуємо новий стагічний метод makeEvent().....	421
Версія друга: реєструємо і отримуємо ControllerEvents.....	422
Вправи і головоломки.....	426

## 13

## Попрацюйте над Swing

Компоненти на  
північі та півдні  
отримують бажану  
висоту.



Компоненти на заході  
та сході отримують  
бажану ширину.

**Swing** — це просто. Доти, доки вам байдуже, де ви опинитесь. Код для роботи зі *Swing* виглядає просто, але, скомпілювавши його, запустивши і подивившись на екран, ви подумаете: «Агов, цей об'єкт має бути в іншому місці!» Інструмент, що *спрощує створення коду і водночас ускладнює управління ним*, — це **менеджер макету інтерфейсу (Layout Manager)**. Проте, доклавши невеликих зусиль, ви зможете підкорити його. У цьому розділі ми будемо працювати зі *Swing* і ближче познайомимося з віджетами.

Компоненти <i>Swing</i> .....	432
Менеджери компоновання (Layout managers) .....	433
Як менеджер компоновання ухвалює рішення .....	434
Три головні менеджери компоновання: border, flow та box ....	435
Пограємося зі <i>Swing</i> -компонентами .....	445
Створюємо <i>BeatBox</i> .....	451
Вправи і головоломки .....	456

## 14

## Збереження об'єктів

**Об'єкти можуть бути стиснутими та відновленими.** Вони характеризуються станом і поведінкою. Поведінка міститься в класі, а стан визначається кожним об'єктом окремо. Якщо вашій програмі необхідно зберегти стан, ви можете зробити це складним способом, опитуючи кожен об'єкт і старанно записуючи значення кожної змінної екземпляра. Або ви можете **зробити це легким ОО-способом** — просто застосуйте суху заморозку (серіалізуйте об'єкт) та відновіть (десеріалізуйте), аби отримати його знов.

Серіалізований



Маєте питання?



Десеріалізований

Лови ритм .....	462
Збереження стану .....	463
Запис серіалізованого об'єкта у файл .....	464
Що насправді відбувається з об'єктом під час серіалізації .....	466
Але що конкретно є станом об'єкта? Що слід зберігати? .....	467
Інтерфейс <i>Serializable</i> .....	469
Десеріалізація: відновлення об'єкта .....	473
Збереження і відновлення ігрових персонажів .....	476
Клас <i>GameCharacter</i> .....	477
Серіалізація об'єктів .....	478
Запис рядка в текстовий файл .....	479
Приклад текстового файлу: електронні флеш-карти .....	480
<i>Quiz Card Builder</i> , структура коду .....	481
Клас <i>java.io.File</i> .....	484
Читання з текстового файлу .....	486
<i>Quiz Card Player</i> , структура коду .....	487
Розбір тексту за допомогою методу <i>split()</i> із класу <i>String</i> .....	490
Ідентифікатор версії: «великий капкан» серіалізації .....	492
Використання <i>serialVersionUID</i> .....	493
Збереження патерну <i>BeatBox</i> .....	495
Відновлення схеми <i>BeatBox</i> .....	496
Вправи і головоломки .....	498



## Ми вважаємо читача «Head First» учнем

Яким чином ми щось *дізнаємося*? Спочатку потрібно це «щось» *зрозуміти*, а потім *не забути*. Заштовхати в голову безліч фактів недостатньо. Згідно з даними новітніх досліджень в галузі когнітивістики, нейробіології та психології навчання, для засвоєння матеріалу потрібно щось більше, ніж текст на сторінці. Ми знаємо, як змусити ваш мозок працювати.



### Основні принципи серії «Head First»:

**Зробіть це наочним.** Графіка запам'ятовується значно краще за звичайний текст і значно підвищує ефективність сприйняття інформації (до 89 % підвищується пригадування та відтворення вивченого). До того ж, матеріал стає зрозумілішим. Якщо **розташувати відповідний текст на малюнках або поряд з ними**, а не під ними або на сусідній сторінці, учні отримують удвічі більше шансів зрозуміти зміст і вирішити проблеми, які описуються.



Погоно бути абстрактним методом. Докондяться обходиться без тіла.



```
abstract void roam();
```

Метод не має тіла! Не забудьте поставити крапку з комою.

**Використовуйте розмовний та персоналізований стиль викладу.** Нещодавні дослідження показали, що при особистісному (направленому на читача напряму) розмовному стилі викладу матеріалу, замість формальних лекцій поліпшення результатів на підсумковому тестуванні складало до 40 %. Розповідайте історію замість того, щоби читати лекцію. Не ставтеся до себе занадто серйозно. Що ймовірніше приверне вашу увагу: цікава бесіда за вечерею або лекція?

Чи можна сказати, що ванна є (IS-A) ванною кімнатою? Ванна кімната є (IS-A) ванною? Або це взаємопов'язані (HAS-A) речі?



**Змусьте учня мислити глибше.** Доки ви не почнете ворухити звивинами, у вашій голові нічого не відбуватиметься. Читач повинен бути зацікавлений у результаті; він повинен натхненно вирішувати завдання, формулювати висновки і генерувати нові знання. А для цього необхідні вправи і каверзані питання, у пошуку відповідей на які задіяні обидві півкулі мозку й різноманітні почуття.

**Приверніть увагу читача — і зафіксуйте її.** Ситуація, знайома кожному: «Я дуже хочу вивчити це, але засинаю вже після першої сторінки». Мозок звертає увагу на цікаве, дивне, привабливе, несподіване. Вивчення складної технічної теми не має бути нудним. Ваш мозок осягне це набагато швидше, якщо це буде цікаво викладено.

**Залучення емоцій.** Відомо, що наша здатність запам'ятовувати значною мірою залежить від емоційного співпереживання. Ми запам'ятовуємо те, що нам не байдуже. Ми запам'ятовуємо, коли щось відчуваємо. Ні, ми не маємо на увазі хвилюючі історії про хлопчика та його собаку. Мова йде про такі емоції, як здивування, цікавість, інтерес і відчуття власної значущості при вирішенні задачі, яку оточення вважає надто складною, або коли ви зрозуміли, що розбираєтеся в темі краще за всезнайку Боба з технічного відділу.

