1. Як працює лічильник у CountDownLatch і як він впливає на блокування потоків?
A. Лічильник збільшується при кожному виклику await()
B. Потоки блокуються, доки лічильник не стане від'ємним
C. Потоки блокуються, доки лічильник не досягне нуля
D. Лічильник автоматично скидається після розблокування
Правильна відповідь: C
Пояснення:
CountDownLatch ініціалізується значенням лічильника. Потоки, що викликають await(), блокуються до моменту, поки інші потоки не зменшать лічильник до нуля через countDown().

2. У чому полягає можливість перевикористання (cyclic) бар'єру в CyclicBarrier?

A. Бар'єр автоматично створює новий об'єкт після кожного використання
B. Бар'єр може бути використаний повторно після досягнення всіма потоками точки синхронізації
C. Бар'єр дозволяє динамічно змінювати кількість потоків
D. Бар'єр працює лише один раз

Правильна відповідь: B
Пояснення:
Після того як усі потоки досягають бар'єру, він скидається і може бути використаний знову в наступному циклі синхронізації.

3. Як Phaser підтримує динамічне додавання та видалення «сторін» (participants)?

A. Через автоматичний підрахунок активних потоків JVM
B. За допомогою методів register() та arriveAndDeregister()
C. Лише під час створення об'єкта
D. Тільки через спадкування класу Phaser

Правильна відповідь: B
Пояснення:
Phaser дозволяє динамічно реєструвати нових учасників (register) і видаляти їх після завершення (arriveAndDeregister), що робить його дуже гнучким.

4. Які методи CompletableFuture використовуються для запуску асинхронних задач?

A. thenApply(), thenAccept()
B. runAsync(), supplyAsync()
C. get(), join()
D. complete(), cancel()
Правильна відповідь: B
Пояснення:
runAsync() запускає задачу без результату,
supplyAsync() — із поверненням результату, обидва методи виконують задачу асинхронно.

5. Які основні методи класу ReentrantLock забезпечують керування блокуванням?

A. wait(), notify()
B. lock(), unlock(), tryLock(), lockInterruptibly()
C. synchronize(), release()
D. enter(), exit()
Правильна відповідь: B
Пояснення:
ReentrantLock надає явне керування блокуванням, включно з можливістю неблокуючої спроби (tryLock) і перериваного блокування.

6. Яке призначення абстрактного класу ForkJoinTask і які основні методи він надає?

A. Для створення звичайних потоків
B. Для керування пулом потоків
C. Для представлення задач у ForkJoinPool
D. Для синхронізації доступу до ресурсів

Правильна відповідь: C
Пояснення:
ForkJoinTask — базовий клас для задач у ForkJoinPool. Він надає методи fork(), join(), invoke() для паралельного виконання.

7. У чому полягає відмінність між RecursiveTask<V> та RecursiveAction?

A. RecursiveTask не підтримує рекурсію
B. RecursiveAction повертає результат
C. RecursiveTask повертає результат, RecursiveAction — ні
D. Вони повністю ідентичні
Правильна відповідь: C
Пояснення:
RecursiveTask<V> використовується, коли потрібен результат,
RecursiveAction — для задач без повернення значення.

8. Яке призначення методів shutdown() та shutdownNow() і чим вони відрізняються?
A. Обидва негайно зупиняють всі потоки
B. shutdown() чекає завершення задач, shutdownNow() намагається перервати їх
C. shutdownNow() працює лише для SingleThreadExecutor
D. Вони ідентичні
Правильна відповідь: B
Пояснення:
shutdown() — коректне завершення (graceful),
shutdownNow() — примусове, з поверненням списку невиконаних задач.

9. Яке призначення методу getState() і яку інформацію він повертає?

A. Повертає пріоритет потоку
B. Повертає ідентифікатор потоку
C. Повертає поточний стан потоку (Thread.State)
D. Повертає інформацію про стек викликів
Правильна відповідь: C
Пояснення:
getState() повертає стан потоку: NEW, RUNNABLE, BLOCKED, WAITING, TIMED_WAITING, TERMINATED.

10. У яких сценаріях варто застосовувати SingleThreadExecutor і які гарантії він забезпечує?

A. Для максимального паралелізму
B. Для виконання задач у довільному порядку
C. Для послідовного виконання задач в одному потоці
D. Для виконання лише однієї задачі
Правильна відповідь: C

Пояснення:
SingleThreadExecutor гарантує, що всі задачі виконуються послідовно в одному потоці, зберігаючи порядок надходження.