Національний технічний університет України
«КИЇВСЬКИЙ ПОЛІТЕХНІЧНИЙ ІНСТИТУТ
імені ІГОРЯ СІКОРСЬКОГО»

Кафедра інженерії програмного забезпечення в енергетиці

АСИНХРОННЕ ПРОГРАМУВАННЯ

Робоча програма навчальної дисципліни (Силабус)

Реквізити навчальної дисципліни	
Рівень вищої освіти	**Перший (бакалаврський)**
Галузь знань	*12 Інформаційні технології*
Спеціальність	*121 Інженерія програмного забезпечення*
Освітня програма	*Інженерія програмного забезпечення інтелектуальних кібер-фізичних систем в енергетиці*
Статус дисципліни	*Професійної та практичної підготовки (за вибором студентів)*
Форма навчання	*Очна (денна)*
Рік підготовки, семестр	*3 курс, 5 семестр*
Обсяг дисципліни	*4 кредити/120 годин, з яких 54 години аудиторних (36 год. лекції, 18 год. Практичні), (66 годин становить самостійна робота)*
Семестровий контроль/ контрольні заходи	*Усний залік, МКР*
Розклад занять	*http://rozklad.kpi.ua*
Мова викладання	*Українська*
Інформація про керівника курсу / викладачів	*Лектор: к.т.н., Єрошкін Юрій Миколайович*
Розміщення курсу	Засоби Google де викладені матеріали: Лекції, Практики, Лабораторні, Домашні завдання, Література: https://drive.google.com/drive/folders/1N87bvVckuWsIVvwWFdc0hkccciKYNtJt

Програма навчальної дисципліни

1. Опис навчальної дисципліни, її мета, предмет вивчення та результати навчання

Силабус навчальної дисципліни «Асинхронне програмування» (ПВ 04 Ф-Каталогу) складено відповідно до освітньої програми «Інженерія програмного забезпечення інтелектуальних кібер-фізичних систем в енергетиці» підготовки бакалаврів спеціальності 121 – Інженерія програмного забезпечення.

Метою навчальної дисципліни *є формування та закріплення у студентів наступних здатностей: Здатність розробляти архітектури, модулі та компоненти програмних систем (ФК 3); Здатність реалізувати фази та ітерації життєвого циклу програмних система інформаційних технологій на основі відповідних моделей і підходів розробки програмного забезпечення (ФК 11); а також фундаментальних основ розробки багатопоточних програм на прикладі мови програмування Java за рахунок обґрунтованого та усвідомленого застосування різних примітивів синхронізації, законів публікації стану змінних та відстеження змін їх контексту, знайомство з моделлю акторів та її застосуванням в високонавантажених серверних аплікаціях.*

Предмет навчальної дисципліни *– фундаментальні основи розробки багатопоточних програм на прикладі мови програмування Java.*

Програмні результати навчання, на формування та покращення яких спрямована дисципліна: *(ПРН 12) Застосовувати на практиці ефективні підходи щодо проектування програмного забезпечення; (ПРН 15) Мотивовано обирати мови програмування та технології розробки для розв'язання завдань створення і супроводження програмного забезпечення; (ПРН 28) Володіти методами та засобами створення мобільних додатків, крос- та мульти-платформного програмування, зокрема, для кібер-фізичних систем; (ПРН 33) Вміти*

створювати програмне забезпечення для інтелектуальних кібер-фізичних систем, в тому числі з врахуванням специфіки предметної області енергетичної галузі.

2. **Пререквізити та постреквізити дисципліни (місце в структурно-логічній схемі навчання за відповідною освітньою програмою)**

Дисципліна «Асинхронне програмування» для підготовки бакалаврів зі спеціальності 121 Інженерія програмного забезпечення складена на основі освітньої програми «Інженерія програмного забезпечення інтелектуальних кібер-фізичних систем в енергетиці» та навчального плану кафедри інженерії програмного забезпечення в енергетиці НН ІАТЕ.

У структурно-логічній схемі навчання дисципліна «Асинхронне програмування» розміщена тоді, коли студенти вже прослухали навчальні дисципліни з «Алгоритми та структури даних» (ПО 01), «Основи програмування» (ПО 02), що достатньо для виконання практичних робіт з даної дисципліни.

Дисципліна «Асинхронне програмування» забезпечує підготовку до вивчення «Розробка програмного забезпечення мобільних пристроїв» (ПО 22), «Переддипломна практика (ПО 10) та «Дипломне проектування» (ПО 11), які викладаються пізніше.

3. **Зміст навчальної дисципліни**

Розділ 1. Основи конкурентного програмування в Java

1) Модель пам'яті Java (Java Memory Model).
- Принципи видимості змінних, volatile, "happens-before", атомарність.

2) Потоки виконання та управління багатозадачністю.
- Thread, Runnable, синхронізація, synchronized, wait/notify.

3) Конкурентні структури даних.
- ConcurrentHashMap, BlockingQueue, CopyOnWriteArrayList.
- Атомарні змінні: AtomicInteger, AtomicReference.

Розділ 2. Високорівневі абстракції конкурентного виконання

1) Thread Pools та Executors.
- ExecutorService, ScheduledExecutorService, ForkJoinPool.

2) Управління задачами.
- Callable, Future, CompletionService.

3) Асинхронне виконання задач з пріоритетами та таймаутами.
- Використання ExecutorCompletionService, обробка винятків.

Розділ 3. CompletableFuture – асинхронне програмування без блокування

1) Callback-механізми.
- Синхронні та асинхронні зворотні виклики, chaining.

2) CompletableFuture.
- thenApply, thenCompose, handle, exceptionally, allOf, anyOf.
- Безблокувальна обробка результатів, паралельне виконання задач.

3) Композиція, винятки, планування.
- Створення асинхронних сервісів на основі CompletableFuture.

Розділ 4. Асинхронність у Spring Framework

1) Асинхронні сервіси з @Async.
- Налаштування @EnableAsync, створення асинхронних методів.
- AsyncConfigurer, визначення кастомного Executor.

2) Обробка результатів через Future та CompletableFuture.
- Інтеграція з бізнес-логікою, обробка винятків.

3) Асинхронні події та обробники.
- ApplicationEventPublisher, @Async listeners.
- Приклади реакції на події без блокування основного потоку.

4) Spring WebFlux (огляд).
- Нетрадиційний підхід до реактивного програмування у Spring.

- Порівняння Spring MVC vs Spring WebFlux (без занурення у Project Reactor).

Розділ 5. Асинхронні I/O операції

1) **Java NIO.**
   - Буфери, канали, селектори, неблокуючий ввід/вивід.
2) **Асинхронні канали (AsynchronousChannel).**
   - AsynchronousSocketChannel, AsynchronousFileChannel, CompletionHandler.
3) **Асинхронне обслуговування клієнтів.**
   - Побудова неблокуючих серверів та клієнтів (експерименти з NIO.2).

Розділ 6. Реактивне програмування в Java

1) **Основи реактивного підходу (Reactive Manifesto).**
   - Асинхронність, потоковість, відмовостійкість, масштабованість.
2) **RxJava та Reactor Core.**
   - Основні типи: Flux, Mono, Observable, Flowable.
   - Оператори трансформації та комбінування: map, flatMap, retry, zip.
3) **Backpressure та ефективна обробка потоку подій.**
   - Управління навантаженням, пропускною здатністю.

Розділ 7. Акторна модель в Java з Akka

1) **Основи акторної моделі.**
   - Принципи Actor Model, поштові скриньки, стан як повідомлення.
2) **Akka для Java.**
   - Побудова акторних дерев, fault tolerance, supervisor strategies.
3) **Інтеграція Akka у Java-проєкти.**
   - Асинхронні сценарії: розподілені черги, таймери, обробка подій.

Розділ 8. Шаблони та проєктування асинхронних систем

1) **Асинхронні шаблони проєктування.**
   - Promise, Reactor, Retry, Timeout, Event-driven, Scheduler, Producer–Consumer та Scatter–Gather.
2) **Моделювання поведінки систем.**
   - Побудова FSM, обробка сценаріїв на основі подій.
3) **Кращі практики та оптимізація.**
   - Аналіз продуктивності, логування потоків, профілювання, уникнення помилок проектування.

4. Навчальні матеріали та ресурси

Основна література

1. Reactive Java Programming - Andrea Maglie , 4 novembre 2016, 128 pages

2. Reactive Programming with RxJava: Creating Asynchronous, Event-Based Applications, Erik Meijer (Author), Tomasz Nurkiewicz (Author), Ben Christensen (Author), O'Reilly Media, November 22, 2016, 370 pages

3. Modern Java in Action: Lambdas, streams, functional and reactive programming 2nd Edition, by Raoul-Gabriel Urma (Author), Mario Fusco (Author), Alan Mycroft (Author), Manning, November 15, 2018, 592 pages

4. Akka in Action, Second Edition, Francisco Lopez-Sancho Abraham, Manning, June 2023, ISBN 9781617299216, 400 pages

5. Guide to Java Concurrency. – [Електронний ресурс.] – Режим доступу: https://howtodoinjava.com/series/java-concurrency/

6. Java Concurrency. – [Електронний ресурс.] – Режим доступу: https://www.baeldung.com/java-concurrency

7. Brian Goetz. Java Concurrency in practice. – [Електронний ресурс.] – Режим доступу: https://leon-wtf.github.io/doc/java-concurrency-in-practice.pdf

8. Akka: Documentation. – [Електронний ресурс.] – Режим доступу: https://akka.io/docs/

5. Методика опанування навчальної дисципліни(освітнього компонента).

Лекційні заняття

№ з/п	Лекція та її опис
1	Лекція 1. **Модель пам'яті Java. Проблеми видимості та синхронізації:** • Поняття Java Memory Model (JMM) та порядок виконання інструкцій. • Проблеми видимості змінних у багатопотоковому середовищі. • Використання volatile, synchronized і правила «happens-before».
2	Лекція 2. **Потоки в Java: Thread, Runnable, синхронізація:** • Створення та запуск потоків за допомогою Thread та Runnable. • Синхронізація потоків за допомогою synchronized. • Взаємодія потоків: wait(), notify(), notifyAll().
3	Лекція 3. **Конкурентні структури даних і атомарні змінні:** • ConcurrentHashMap, CopyOnWriteArrayList, BlockingQueue. • Пакет java.util.concurrent.atomic: AtomicInteger, AtomicReference. • Гарантії атомарності та thread-safe доступ до спільних даних.
4	Лекція 4. **Використання ExecutorService, Callable, Future:** • Основи роботи з ExecutorService та пули потоків. • Інтерфейси Callable і Future для обробки результатів. • Завершення, скасування і обробка винятків у потоках.
5	Лекція 5. **Основи CompletableFuture:** • Створення асинхронних обчислень за допомогою CompletableFuture. • Використання методів thenApply, thenAccept, handle. • Обробка винятків через exceptionally та логування помилок.
6	Лекція 6. **Комбінування задач з CompletableFuture:** • Зв'язування та послідовність виконання за допомогою thenCompose. • Паралельне виконання: allOf, anyOf, приклади оптимізації. • Реалізація складних бізнес-процесів через ланцюги обчислень.
7	Лекція 7. **Шаблони Producer–Consumer та Scatter–Gather:** • Використання черг у шаблоні Producer-Consumer (BlockingQueue). • Сценарії фан-аут/фан-ін в шаблоні Scatter-Gather (invokeAll, CompletableFuture.allOf). • Застосування шаблонів у потоковій обробці та мікросервісах.
8	Лекція 8. **Асинхронне виконання у Spring: @Async, ExecutorService, CompletableFuture:** • Активація асинхронного режиму: @EnableAsync, @Async. • Використання Executor, AsyncConfigurer для кастомних пулів. • Інтеграція CompletableFuture в Spring-сервіси та обробка винятків.

9	Лекція 9. **Планування задач у Spring: @Scheduled, ScheduledExecutorService:** • Розклад виконання задач за допомогою @Scheduled. • Фіксований інтервал, затримка, cron-вирази. • Альтернативи: ScheduledExecutorService, кастомні планувальники.
10	Лекція 10. **Асинхронна обробка подій у Spring:** • Впровадження подій через ApplicationEventPublisher. • Створення асинхронних слухачів з @EventListener + @Async. • Побудова реактивних реакцій на внутрішні події додатку.
11	Лекція 11. **Java NIO (New I/O):** • Робота з буферами та їх використання в контексті асинхронного програмування. • Використання селекторів для моніторингу подій на каналах. • Реалізація асинхронного читання та запису з використанням Java NIO.
12	Лекція 12. **Asynchronous I/O в Java:** • Використання AsynchronousChannel та CompletionHandler для асинхронного вводу/виводу. • Розуміння основ асинхронних операцій з файловою системою. • Обробка реальних сценаріїв використання асинхронного вводу/виводу.
13	Лекція 13. **Реактивне програмування у Java: RxJava, Reactor, WebFlux (огляд):** • Основи реактивного підходу: Reactive Streams, backpressure. • Типи Mono, Flux, Observable, Flowable. • Коли використовувати Spring WebFlux: порівняння з Spring MVC.
14	Лекція 14. **Оптимізація реактивного коду:** • Дослідження методів оптимізації реактивного коду. • Використання інструментів для виявлення та вирішення проблем продуктивності. • Оптимізація керування ресурсами та уникнення гарячих точок.
15	Лекція 15. **Вступ до акторної моделі. Основи Akka у Java:** • Концепція акторів як окремих обчислювальних одиниць. • Модель ActorSystem у фреймворку Akka. • Переваги використання акторів у паралельних і розподілених системах.
16	Лекція 16. **Акторська модель та паралельність в Akka:** • Розгляд використання акторської моделі для паралельного та асинхронного програмування. • Взаємодія акторів у розподіленій системі та управління станом.
17	Лекція 17. **Інтеграція Akka в додатки:** • Розробка та інтеграція асинхронних компонентів на базі Akka в реальних додатках. • Використання Akka для створення розподілених систем та обробки великого обсягу асинхронних подій.

18	Лекція 18. **Вивчення кращих практик та оптимізацій:** • Аналіз кращих практик асинхронного програмування в широкому контексті. • Використання інструментів моніторингу та профілювання для виявлення можливостей оптимізацій. • Розгляд та адаптація оптимальних підходів в реальних проектах.

Практичні заняття

№ з/п	Практичні робота та їх опис
1	Запрограмувати роботу 2-х світлофорів на перехресті. **Опис**: Перемикання світла на одному спричинює відповідне перемикання світла на іншому. Реалізація кожного світлофора як окремого потоку. Перемикання світла реалізувати за допомогою notify() чи notifyAll(). Встановити середовище розробки для МП Java. Створення потоків за допомогою класу Thread та інтерфейсу Runnable. Використання synchronized методів.
2	Реалізувати пішохода, який викликатиме запит перемикання світлофора на зелений для нього. Всі події логувати, наприклад, за допомогою log4j. Практика CompletableFuture. **Опис**: До минулої практичної додати можливість кнопки-перемикання світлофора для людей, яким треба перейти дорогу. Кнопку реалізувати за допомогою CompletableFuture. Логувати всі перемикання світлофора з описом, хто причина перемикання (кнопка чи сам світлофор).
3	Зчитати та записати інформацію в файл за допомогою Java NIO. **Опис**: Запис логів перевести у файл та реалізувати за допомогою Java NIO. Також зробити виведення на екран інформації файлу через зчитування його за допомогою Java NIO.
4	За допомогою RxJava написати роботу 2-х світлофорів користуючись шаблоном спостерігача. **Опис**: Створіть Observable, виведіть дані та підпишіться на нього. Реалізуйте, процес створення подій та їхнє зчитування спостерігачами - іншими світлофорами на перехресті та пішоходами, котрі хочуть перейти дорогу натиснувши на кнопку. В роботі необхідно використати методи map, filter, flatMap. Розгляньте обробку помилок в RxJava. Вивчіть, як використовувати оператори, такі як onErrorResumeNext, onErrorReturn, retry, для управління помилками. Експериментуйте з паралельною обробкою подій за допомогою observeOn та subscribeOn. Розумійте, як RxJava може використовувати різні потоки для виконання різних частин коду.
5	Переписати роботу зі світлофорами за допомогою Akka. **Опис**: Встановіть Akka. Використовуйте системи управління залежностями, такі як Maven або Gradle, для додавання Akka до вашого проекту. Створіть Акка акторів для світлофорів та пішоходів. Скористайтесь класами акторів. Ознайомтесь із методами обробки помилок та відновлення стану в Akka, такими як supervisorStrategy.
6	Виконати рефакторинг вибраної практичної за найкращими практиками написання ООП та асинхронного коду. **Опис**: користуючись поняттями Clean Code та шаблони проектування ПЗ, переробити вибрану практичну.

6. Самостійна робота студента

№ з/п	*Назва теми, що виноситься на самостійне опрацювання*
1	Тема 1. Асинхронність з спільною пам’яттю
2	Тема 2: Безпека потоків (Thread safety)
3	Тема 3: Спільний доступ до об’єктів (Sharing Objects)
4	Тема 4: Проектування об’єктів (Composing Objects)
5	Тема 5: Виконання задач (Task execution)
6	Тема 6: Зупинка та скасування виконання задач (Cancellation and Shutdown)
7	Тема 7: Одночасне обслуговування багатьох потоків (Applying Thread Pools)
8	Тема 8: Уникнення загроз живучості (Avoiding Liveness Hazards)
9	Тема 9: Продуктивність та масштабованість (Performance and Scalability)
10	Тема 10: Кероване управління доступом (Explicit Locks)
11	Тема 11: Розробка примітивів синхронізації (Building Custom Synchronizers)
12	Тема 12: Атомарні змінні та неблокуюча синхронізація (Atomic Variables and Non-blocking Synchronization)
13	Тема 13: Модель пам’яті віртуальної машини Java (The Java Memory Model)
14	Тема 14. Асинхронність з ізольованою пам’яттюТема
15	Тема 15: Фреймворк моделі акторів Akka (Akka Framework)
16	Тема 16: Життєвий цикл актора в фреймворку Akka (Actor life-cycle)
17	Тема 17: Комунікація акторів та управління ними (Actor Communication and Supervising)
18	Тема 18. Асинхронна робота з базами даних

Додаткові практичні роботи

№ з/п	Практична робота та її опис
1	Використання змінних volitile та ThreadLocal.
2	Використання патерну singleton в потоках. Використання Thread.lock/unlock або Atomic замість synchronized методів.
3	Використання екзекютор сервісів для створення потоків.
4	Демонстрація роботи Interrupt для потоків та Shutdown для екзекютор сервісів.
5	Використання методів переривання wait(),notify()/notifyAll().

з/п	Вид самостійної роботи	Кількість годин СРС
1	Опрацювання тем, які винесені на самостійну роботу	30
2	Виконання практичних робіт	20
3	Підготовка до МКР	10
4	Підготовка до заліку	6

Політика та контроль

7. Політика навчальної дисципліни (освітнього компонента)

Для успішного проходження курсу та складання контрольних заходів необхідним є вивчення навчального матеріалу за кожною темою. Специфіка курсу передбачає акцент на розумінні підходів і принципів, отримання практичних навичок, а не просто запам'ятовування визначень. Кожен студент повинен ознайомитися і слідувати Положенню про дистанційне навчання в КПІ ім. Ігоря Сікорського (https://osvita.kpi.ua/node/188), Положенню про систему оцінювання результатів навчання (https://osvita.kpi.ua/node/37), Положенню про поточний, календарний та семестровий контроль результатів навчання (https://osvita.kpi.ua/node/32), які унормовують форми контрольних заходів та критеріїв оцінювання навчальних досягнень здобувачів вищої освіти в КПІ ім. Ігоря Сікорського, а також ознайомитися з нормативно-правовим та регламентуючими документами й корисними ресурсами з розвитку культури академічної доброчесності та запобігання плагіату в КПІ ім. Ігоря Сікорського https://kpi.ua/academic-integrity. Для успішного засвоєння програмного матеріалу студент зобов'язаний:

– не запізнюватися на заняття;

– не пропускати заняття, а в разі пропуску відновити за допомогою консультування з викладачем та з використанням Classroom/Кампус конспект, самостійно вивчити матеріал пропущеного заняття та скласти відповідні контрольні заходи в індивідуальному порядку;

– конструктивно підтримувати зворотній зв'язок на всіх заняттях;

– брати активну участь у освітньому процесі;

– своєчасно і старанно виконувати завдання для самостійної роботи;

– бути доброзичливим до однокурсників та викладачів;

– брати участь у контрольних заходах;

– за об'єктивних причин (наприклад, хвороба, міжнародне стажування) навчання може відбуватись індивідуально (в дистанційній online формі за погодженням із директором інституту);

– будь-яке копіювання або відтворення результатів чужої праці (у тому числі списування), якщо тільки робота не має груповий формат, використання чужих завантажених з Інтернету матеріалів кваліфікується як порушення норм і правил академічної доброчесності та передбачає притягнення винного до відповідальності, у порядку, визначеному чинним законодавством та Положенням про академічну доброчесність університету. Результатом невиконання та/або недотримання правил може бути оцінка «не зараховано» за курс.

8. Види контролю та рейтингова система оцінювання результатів навчання (РСО)

Поточний контроль: *тестування або експрес-опитування за кожним Розділом навчального матеріалу, Модульна контрольна робота, виконання завдань до практичних занять.*

Календарний контроль: *проводиться двічі на семестр як моніторинг поточного стану виконання вимог Силабусу.*

Модульна контрольна робота складається *з тесту за матеріалом Розділів 1-6.*

Семестровий контроль*: залік.*

Умови допуску до семестрового контролю*: семестровий рейтинг більше 40 балів.*

Система рейтингових (вагових) балів та критерії оцінювання

Максимальна кількість балів з кредитного модуля дорівнює 100.

Рейтинг студента з дисципліни складається з балів, що він отримує за:

- виконання та захист практичних робіт,
- модульну контрольну роботу (МКР) тривалістю 1 акад. година.

Виконання завдань практичних робіт

Завдання практичні роботи являє собою індивідуальне виконання робіт, що пов'язані з рішенням на ЕОМ заданої задачі шляхом розробки модулю і його інтерфейсу. Інтерфейс повинен бути поєднаний з рішеннями практик.

Вагові бали завдань наведено у таблиці.

№ з/п	*Види завдань*	*Внесок до семестрового рейтингу балів*
1	Запрограмувати роботу 2-х світлофорів на перехресті.	15
2	Реалізувати пішохода, який викликатиме запит перемикання світлофора на зелений для нього. Всі події логувати, наприклад, за допомогою log4j. Практика CompletableFuture.	15
3	Зчитати та записати інформацію в файл за допомогою Java NIO.	10
4	За допомогою RxJava написати роботу 2-х світлофорів, користуючись шаблоном спостерігача.	15
5	Переписати роботу зі світлофорами за допомогою Akka.	15
6	Виконати рефакторинг вибраної практичної за найкращими практиками написання ООП та асинхронного коду.	10
7	МКР	20

Максимальна кількість балів за всі завдання дорівнює 100 балів.

Критерії оцінювання

Підготовка до роботи (у відсотках від максимальної кількості балів за відповідну роботу):

- якщо робота виконана не самостійно та простежується не індивідуальне виконання то знімається 50% від максимальної кількості балів;
- якщо в програмі не витримані основні правила створення програмних продуктів (модульність, дружній інтерфейс, наявність коментарів та т.п.) знімається 5%.

Виконання завдання практичної роботи:

- робота виконана повністю і вірно протягом відведеного часу – 50 %;

– робота виконана пізніше зазначеного терміну – 20 %;

Якість захисту роботи:

– студент вірно і повністю відповів на запитання – 30 %;
– студент при відповіді допустив несуттєві неточності – 20 %;
– студент при відповіді на запитання допустив суттєві неточності, але самостійно виправив їх – 10 %.

2. Модульна контрольна робота

Ваговий бал – 20.

Модульна контрольна робота складається з 10 тестових завдань. За кожну вірну відповідь на запитання надається 2 бали.

Сума вагових балів контрольних заходів протягом семестру складає:

R =80 (практичні) 20 (МКР) = 100 балів.

Умови допуску до семестрового контролю*: зарахування усіх практичних робіт та семестровий рейтинг не менше 40 балів.*

Таблиця відповідності рейтингових балів оцінкам за університетською шкалою:

Бали (RD)	Традиційна оцінка
95..100	Відмінно
85…94	Дуже добре
75…84	Добре
65…74	Задовільно
60…64	Достатньо
RD<=60	Незадовільно
RD < 40 або не виконані інші умови допуску до заліку	Не допущений

Форма семестрового контролю – залік

Максимальна сума балів складає 100. Необхідною умовою допуску до заліку є зарахування всіх практичних робіт та. Для отримання заліку з кредитного модулю «автоматом» потрібно мати рейтинг не менше 60 балів, а також виконані умови допуску до заліку.

Здобувачі, які наприкінці семестру мають рейтинг менше 60 балів, а також ті, хто хоче підвищити свою оцінку в системі ECTS, виконують залікову контрольну роботу. При цьому набрані бали здобувачем анулюються, а оцінка за залікову контрольну роботу є остаточною.

Залікова робота. *Залікова робота проводиться на останньому лекційному занятті. Здобувач проходить тестування очно або у середовищі дистанційного навчання, наприклад Moodle. На тестування пропонується 100 тестових питань, кожне з яких оцінюється в 1 бал. Для отримання позитивної оцінки необхідно набрати 60 балів і вище. Час тестування зазвичай складає 100 хвилин, але може бути скоригований лектором та (або) викладачам, що приймає залік.*

9. Додаткова інформація з дисципліни (освітнього компонента)

Перелік питань, які виносяться на семестровий контроль:

- *ООП в JavaПотоки та Багатозадачність*
- *Concurrency API та Executors*
- *Callback-функції*
- *CompletableFuture*
- *Реактивне програмування*
- *Java NIO (New I/O)*
- *Asynchronous I/O в Java*

- *Асинхронний код*
- *Застосування Reactor або RxJava*
- *Реактивні потоки та операції*
- *Оптимізація реактивного коду*
- *Принципи Akka:*
- *Акторська модель та паралельність в Akka:*
- *Інтеграція Akka в додатки:*
- *Асинхронні шаблони проектування:*
- *Моделювання асинхронних систем:*
- *Основні кращі практики та оптимізацій:*
- *Асинхронність з спільною пам’яттю*
- *Безпека потоків (Thread safety)*
- *Спільний доступ до об’єктів (Sharing Objects)*
- *Проектування об’єктів (Composing Objects)*
- *Виконання задач (Task execution)*
- *Зупинка та скасування виконання задач (Cancellation and Shutdown)*
- *Одночасне обслуговування багатьох потоків (Applying Thread Pools)*
- *Уникнення загроз живучості (Avoiding Liveness Hazards)*
- *Продуктивність та масштабованість (Performance and Scalability)*
- *Кероване управління доступом (Explicit Locks)*
- *Розробка примітивів синхронізації (Building Custom Synchronizers)*
- *Атомарні змінні та неблокуюча синхронізація (Atomic Variables and Non-blocking Synchronization)*
- *Модель пам’яті віртуальної машини Java (The Java Memory Model)*
- *Асинхронність з ізольованою пам’яттюТема*
- *Модель акторів (Actor Model)*
- *Фреймворк моделі акторів Akka (Akka Framework)*
- *Життєвий цикл актора в фреймворке Akka (Actor life-cycle)*
- *Комунікація акторів та управління ними (Actor Communication and Supervising)*
- *Асинхронна робота з базами даних*

Вимоги до спеціального матеріально-технічного та/або інформаційного забезпечення:

Наявність діючих облікових записів: Користувача на Платформі дистанційного навчання "Сікорський" та Сервісів Google;

Інтегроване середовище розробки: з підтримкою мови програмування Java;

Вимоги до мережевої інфраструктури: достатні для отримання доступу до https:// google.com та https://do.ipo.kpi.ua.

Операційна система: не специфікується;

Інтернет браузер: не специфікується;

Текстовий редактор: не специфікується;

Робочу програму навчальної дисципліни (Силабус): Асинхронне програмування

Складено асистентом кафедри інженерії програмного забезпечення в енергетиці НН ІАТЕ, к.т.н. Єрошкіним Юрієм Миколайовичем.

Ухвалено кафедрою інженерії програмного забезпечення в енергетиці НН ІАТЕ(протокол № 42 від 18.06.2025 р.)

Погоджено Методичною комісією НН ІАТЕ (протокол № 9 від 27.06.2025 р.)