**МІЖРЕГІОНАЛЬНА АКАДЕМІЯ УПРАВЛІННЯ ПЕРСОНАЛОМ**

МАУП

# МЕТОДИЧНІ РЕКОМЕНДАЦІЇ ЩОДО ЗАБЕЗПЕЧЕННЯ САМОСТІЙНОЇ РОБОТИ СТУДЕНТІВ

**з дисципліни**

**«Кроссплатформене програмування на мові Java»**
**(для бакалаврів)**



**Київ 2016**

Підготовлено доцентом кафедри комп'ютерних інформаційних систем і технологій Табаковим В.З.

Затверджено на засіданні кафедри комп'ютерних інформаційних систем і технологій
(протокол № 12 від 21.06.2016 р.)

*Схвалено Вченою радою Інституту комп'ютерно-інформаційних технологій* (Протокол № 9 від 22.06. 2016 р.).

**Табаков Валерій Зіновійович** Методичні рекомендації щодо забезпечення самостійної роботи студентів з дисципліни «Кроссплатформене програмування на мові Java» (для бакалаврів спеціальностей 121 Інженерія програмного забезпечення та 122 Комп'ютерні науки та інформаційні технології). – К. : МАУП, 2016. – 24 с.

Методичні рекомендації містять пояснювальну записку, тематичний план, зміст дисципліни, питання для самостійного вивчення студентами та самоконтролю, теми рефератів, питання для самоконтролю та список рекомендованої літератури.

## *ПОЯСНЮВАЛЬНА ЗАПИСКА*

*«Кроссплатформене програмування на мові Java»* є дисципліною вільного вибору студента. Для її вивчення необхідні знання дисциплін «Основи програмування та алгоритмічні мови», «Об'єктно-орієнтоване програмування». Знання, отримані під час вивчення дисципліни будуть використовуватись для підготовки дипломної роботи, та у професійній діяльності.

**Мета дисципліни** *«Кроссплатформене програмування на мові Java»* полягає у вивченні студентами напрямів «Інженерія програмного забезпечення» та «Комп'ютерні науки» основ програмування та створення кроссплатформених додатків на мові *Java*.

**Завдання:** ознайомлення студентів з Java-технологіями, набуття навичок застосування Java-технологій для створення кроссплатформених програмних систем, умінь застосовувати створені за допомогою Java-технологій програмні системи для автоматизації професійної діяльності.

В результаті вивчення дисципліни студент повинен

**знати**

- мову Java, Java- технології;
- середовище розробки Java- додатків Eclipse;
- концепцію об'єктно -орієнтованого програмування в Java;
- структуру Java –додатків;
- класи, інтерфейси і їх ієрархію в Java
- типи, операції та оператори Java;
- потоки, консольне введення –виведення;
- контейнери Java;
- серіалізацию в Java;
- обробку помилок з використанням винятків в Java;
- поняття про компонентні моделі і моделі JavaBeans;
- створення WEB додатків (Сервлети);

**вміти**

- створювати кроссплатформені програмні системи в Java;
- застосовувати створені за допомогою Java-технологій програмні системи для автоматизації професійної діяльності.

Самостійна робота студентів — один з основних засобів оволодіння навчальним матеріалом у час, вільний від аудиторних навчальних занять.

Головною метою самостійної роботи студентів є активізація систематичної роботи студентів, індивідуалізація навчання, підвищення якості засвоєння матеріалу навчальної дисципліни «*Кроссплатформене програмування на мові Java*».

Завдання самостійної роботи студентів — сприяти засвоєнню в повному обсязі знань, умінь, навичок, закріплювати та систематизувати, розширювати та поглиблювати набуті в процесі аудиторної роботи знання, вміння та навички, застосувати їх при виконанні практичних завдань та творчих робіт, а також виявляти прогалини у системі знань із предмету «*Кроссплатформене програмування на мові Java*», закріплювати, а також самостійно вивчати та засвоювати новий матеріал під керівництвом викладача, але без його безпосередньої участі.

Зміст самостійної роботи студента з дисципліни «*Кроссплатформене програмування на мові Java*» визначається навчальною програмою дисципліни, методичними матеріалами, завданнями викладача.

Самостійна робота з навчальної дисципліни «*Кроссплатформене програмування на мові Java*» включає:

- підготовку до практичних занять (для студентів очної форми навчання);
- підготовку до поточного контролю знань студентів з окремих тем навчальної дисципліни (для студентів очної форми навчання);
- підготовку до рубіжного (модульного) контролю (для студентів очної форми навчання);
- письмове виконання контрольних робіт (для студентів заочної форми навчання). Контрольна робота є комплексним завданням, в якому містяться теоретичні і практичні завдання, виконання яких розвиває самостійність при аналітичній обробці теоретичної інформації;
- підготовку до підсумкового контролю знань за контрольними питаннями.

Важливе значення в керівництві самостійною роботою студентів мають індивідуальні та групові консультації, їх мета — допомогти студентам у вивченні того чи іншого питання, в правильній організації самостійної роботи над вивченням предмета. Успішність підготовки до практичних занять і складання іспиту значною мірою залежить від організації самостійної роботи. Для здійснення самостійної роботи студентам

рекомендується ознайомитись з нормативно-правовою базою та навчально-методичною літературою, перелік якої наведено в списку рекомендованої літератури, а також публікаціями періодичних видань.

Рекомендовану літературу необхідно вивчати систематично, згідно зі списком і в такій послідовності:

а) ознайомитись за навчальною програмою зі змістом кожної теми;

б) засвоїти навчальний матеріал, що відноситься до конкретної теми;

в) дати відповіді на питання для самостійної роботи студентів з кожної теми;

г) дати відповіді на контрольні запитання відповідної теми;

д) виписати всі незрозумілі питання для розгляду їх на консультації.

Самостійна робота студента забезпечується системою навчально-методичних засобів, передбачених для вивчення навчальної дисципліни «*Кроссплатформене програмування на мові Java*», підручниками, навчальними та методичними посібниками, методичними матеріалами для самостійної роботи студентів, конспектом лекцій, періодичними виданнями тощо. Самостійна робота студентів з навчальної дисципліни «*Кроссплатформене програмування на мові Java*» організовується з дотриманням низки вимог:

- надання детальних методичних рекомендацій щодо виконання роботи;
- забезпечення можливості творчого підходу у виконанні роботи, не обмежуючи освітній процес виконанням стандартних завдань;
- підтримка у процесі виконання самостійної роботи постійного взаємозв’язку між викладачем та студентами.

Студенти, які розпочинають вивчати дисципліну «*Кроссплатформене програмування на мові Java*», мають бути інформовані викладачем щодо організації самостійної роботи, її форм та видів, термінів виконання, форм контролю та звітності, кількості балів за виконання завдань.

Всі завдання самостійної роботи студентів поділяються на обов’язкові та вибіркові, виконуються у встановлені терміни, з відповідною максимальною оцінкою та передбачають певні форми звітності щодо їх виконання. Обов’язкові завдання виконуються кожним без винятку студентом у процесі вивчення дисципліни, вибіркові завдання є альтернативними. Перелік завдань для самостійної роботи, форми її організації та звітності, термін виконання та кількість отриманих балів за виконані завдання визначаються викладачем кафедри при розробці робочої навчальної програми дисципліни. Оцінки (бали), одержані студентами за виконання різних видів самостійної роботи, фіксуються викладачами і доводяться до відома студентів.

Організація і контроль процесу та змісту самостійної роботи і її результатів здійснюються викладачами кафедри. Основними видами контролю рівня оволодіння навчальним матеріалом студентами очної форми навчання є усне опитування (заочної форми навчання — перевірка контрольних робіт). За результатами контролю студентам виставляють

оцінки в журналах поточної успішності за бальною системою. Підсумковий контроль знань у вигляді заліку здійснюється Підсумковий контроль (іспит) проводиться у вигляді співбесіди за результатами підготовки студентом відповідей на питання екзаменаційних білетів.

## *ІНДИВІДУАЛЬНО-КОНСУЛЬТАЦІЙНА РОБОТА*

Індивідуально-консультативна робота з дисципліни «*Кроссплатформене програмування на мові Java*» організовуються та здійснюється у формі консультацій на кафедрі комп'ютерних інформаційних систем і технологій у відповідності до затвердженого графіку консультацій (одна консультація на два тижні).

На консультаціях студентам надаються пояснення з виконання самостійної роботи, підготовки до практичних занять, перевірка та захист завдань, винесених на поточний контроль тощо.

## *ТЕМАТИЧНИЙ ПЛАН*

***дисципліни*** *"Кроссплатформене програмування на мові Java"*

| № | Назва змістового модуля та теми |
|---|---|
| **Змістовий модуль 1 – Вступ до Java і Java-технологій** | |
| 1. | Введення в Java і Java- технології |
| 2. | Eclipse як середовище розробки Java- додатків |
| **Змістовий модуль 2 – Основи об'єктно-орієнтованого програмування в Java** | |
| 3. | Реалізація концепції об'єктно -орієнтованого програмування в Java |
| 4. | Структура Java –додатків |
| 5. | Класи, інтерфейси і їх ієрархія |
| 6. | Типи Java |
| 7. | Операції та оператори |
| **Змістовий модуль 3 – Засоби Java для створення об’єктно-орієнтованих програмних систем** | |
| 8. | Потоки, консольне введення –виведення |
| 9. | Контейнери Java |
| 10. | Серіалізация в Java |
| 11. | Обробка помилок з використанням винятків |
| 12. | Поняття про компонентні моделі і моделі JavaBeans |
| 13. | Створення WEB додатків (Сервлети) |
| Разом годин: 150 | |

## Зміст
**дисципліни** *" Кроссплатформене програмування на мові Java "*

## Змістовий модуль 1 – Вступ до Java і Java-технологій

***Тема 1. Введення в Java і Java- технології***

Області застосування Java.JDK, JVM. Java 2 (Java EE/SE/ME). Установка JDK/ JRE, настройка параметрів середовища.

Література: [1] - [12].

***Тема 2. Eclipse як середовище розробки Java- додатків***

Середовища розробки . Знайомство з Eclipse як середовищем розробки Java-додатків. Проекти. Основні етапи створення програми в середовищі Eclipse. Найпростіший додаток, компіляція, запуск. Установка Eclipse, перше знайомство з додатком, налагодження.

Література: [1] - [12].

## Змістовий модуль 2 – Основи об'єктно-орієнтованого програмування в Java

***Тема 3. Реалізація концепції об'єктно -орієнтованого програмування в Java***

RTTI, інформація про клас, віртуальні методи. Класи і інтерфейси. Ідеологія Java.

Література: [1] - [12].

***Тема 4. Структура Java –додатків***

Класи і пакети Java і їх співвідношення з елементами файлової системи. Послідовність завантаження класів і вплив її на структуру програми. Стандартні типи та об'єкти Java. Посилання, покажчики і мова

Java. Об'єкти Java, цикл життя об'єктів. Поняття про збір сміття. Архіви Java. Створення простих демонстраційних додатків.

Література: [1] - [12].

***Тема 5. Класи, інтерфейси і їх ієрархія***

Класи, їх структура. Області видимості. Створення ієрархії класів. Перевизначення методів класу. Створення і знищення об'єкта, конструктори. Статичні члени класів. Поточний об'єкт і безпосередній предок поточного об'єкта. Ініціалізація членів класу. Константи, перерахування в Java. Інтерфейси, їх зміст і використання. Інтерфейси і спадкування. Перетворення типів з урахуванням класів і інтерфейсів в умовах поліморфізму. Створення простих додатків (класи, спадкування, інтерфейси, generic Java як система контролю перетворення типів).

Література: [1] - [12].

***Тема 6. Типи Java***

Стандартні типи, їх об'єктні оболонки. Масиви Java. Стандартні типи Java - потоки введення-виведення, рядки, календар і ін.

Створення простих додатків ( введення-виведення, контейнери).

Література: [1] - [12].

***Тема 7. Операції та оператори***

Операції та їх пріоритети. Основні оператори Java, основні прийоми їх використання.

Література: [1] - [12].

## Змістовий модуль 3 – Засоби Java для створення об’єктно-орієнтованих програмних систем

***Тема 8. Потоки, консольне введення –виведення***

Два види об'єктів організації введення-виведення. Введення/виведення з використанням консолі. Файлове введення-виведення. Створення простих додатків (введення-виведення, файли).

Література: [1] - [12].

***Тема 9. Контейнери Java***

Види контейнерів Java. Основні прийоми використання контейнерів Java. Використання шаблонів. Створення простих додатків (контейнери).

Література: [1] - [12].

***Тема 10. Серіалізация в Java***

Роль сериализации в Java. Стандартна процедура сериализации. Поняття про інтерфейс Cloneable.

Література: [1] - [12].

***Тема 11. Обробка помилок з використанням винятків***

Порівняння механізму з використанням винятків з традиційним механізмом обробки помилок. Оброблювані і необроблювані виключення. Стандартні виключення Java- технологій, їх роль. Оператори Java для підтримки винятків.

Література: [1] - [12].

***Тема 12. Поняття про компонентні моделі і моделі JavaBeans***

Використання компонентних моделей при створенні реальних сучасних додатків. Компонентна модель JavaBeans. Властивості, події, дескриптори компонентів. Компоненти JavaBeans і обмін подіями в консольному додатку.

Література: [1] - [12].

***Тема 13. Створення WEB додатків (Сервлети)***

Сервлети Java. Servlet API. Інтерфейс сервлета. Життєвий цикл сервлета. Виклик сервлетов з HTML-сторінки. ServletRequest інтерфейс.

ServletResponse інтерфейс. Читання параметрів сервлета і формування відповіді клієнтові. Читання параметрів ініціалізації. Клас HttpServlet. Робота з HTTP – запитами. Сервер Jakarta Tomcat.

Література: [1] - [12].

## *ЗАВДАННЯ ДЛЯ САМОСТІЙНОЇ РОБОТИ СТУДЕНТІВ*

**До модуля 1**.

Опанувати області застосування Java. Вивчити призначення JDK, JVM. Вивчити призначення Java 2 (Java EE/SE/ME). Здобути навичку установки JDK/ JRE. Здобути навичку настройки параметрів середовища розробки Java-додатків без інтегрованого середовища розробки.

Ознайомитися з фукціональними можливостями середовищ розробки Java-додатків. Опанувати Eclipse як середовище розробки Java-додатків. Опанувати порядок створення Java-проекту. Установити Eclipse. Опанувати основні етапи створення програми в середовищі Eclipse. Створити найпростіший Java-додатокв Eclipse, скомпілювати та запустити його.

**До модуля 2.**

Вивчити призначення RTTI. Дати визначення та приклади класів та інтерфейсів. Викласти ідеологію Java.

Визначити клас та пакети Java. Дати визначення співвідношення класу та пакетів з елементами файлової системи. Навести послідовність завантаження класів та визначити її вплив на структуру програми. Навести стандартні типи та об'єкти Java. Навести визначення посилань та покажчиків у мови Java. Навести визначення об'єктів Java та пояснити цикл життя об'єктів. Навести поняття про збір сміття. Визначити архіви Java.

Навести клас та його структуру. Визначити область видимості класу. Навести порядок створення ієрархії класів. Навести порядок та застосування процедури перевизначення методів класу. Навести порядок створення і знищення об'єкта. Застосування конструктора. Навести визначення статичних членів класу. Навести поняття поточного об'єкту і безпосереднього предку поточного об'єкта. Ініціалізувати член класу. Створити константу в Java. Застосувати інтерфейс. Перетворити тип з урахуванням класів і інтерфейсів в умовах поліморфізму. Створити

простий додаток (з застосуванням класів, спадкування, інтерфейсів та generic Java як системи контролю перетворення типів).

Навести визначення стандартного типу та його об'єктної оболонки. Визначити масив Java. Визначити стандартні типи Java - поток введення-виведення, рядок, календар і ін. Створити простий додаток ( з застосуванням введення-виведення та контейнерів).

Навести операції Java та визначити їх пріоритети. Навести основні оператори Java та основні прийоми їх використання.

**До модуля 3.**

Навести два види об'єктів організації введення-виведення. Запрограмувати введення/виведення з використанням консолі. Запрограмувати файлове введення-виведення. Створити простий додаток з застосуванням консольного та файлового введення-виведення.

Навести види контейнерів Java. Визначити основні прийоми використання контейнерів Java. Навести приклади використання шаблонів. Створити простий додаток з використанням контейнеру.

Визначити роль серіалізації в Java. Навести стандартну процедуру серіалізації. Дати поняття про інтерфейс Cloneable.

Порівняти механізм з використанням винятків з традиційним механізмом обробки помилок. Дати визначення оброблюваних і необроблюваних виключень. Навести стандартні виключення Java-технологій та визначити їх роль. Навести оператори Java для підтримки винятків.

Навести приклади використання компонентних моделей при створенні реальних сучасних Java-додатків. Визначити поняття про компонентну модель JavaBeans. Визначити властивості, події, дескриптори компонентів JavaBeans. Навести приклад застосування компонентів JavaBeans з обміном подіями в консольному додатку.

Визначити сервлет Java. Визначити Servlet API. Визначити Інтерфейс сервлета. Визначити життєвий цикл сервлета. Навести порядок виклику

сервлетів з HTML-сторінки. Визначити ServletRequest інтерфейс. Визначити ServletResponse інтерфейс. Визначити технологію читання параметрів сервлету і формування відповіді клієнтові. Визначити технологію читання ініціалізації сервлету. Визначити клас HttpServlet. Навести приклади роботи з HTTP – запитами. Навести порядок інсталяції та застосування серверу Jakarta Tomcat.

# *ВАРІАНТИ КОНТРОЛЬНИХ РОБІТ*

1. Дан код:

```
class Questl {
      private static void main (String a)
       { System.out.println("Java 2");
      }
      }
```

Які виправлення слід зробити, щоби клас **Quest1** став додатком, що запускається? (виберіть 2 правильні варіанта)

1) Оголосити клас **Quest1** як **public**;
2) Замінити параметр метода **main()** на **String[] a**;
3) Замінити параметр доступа до методу **main()** на **public**;
4) Видалити параметр з оголошення методу **main()** .

2. Виберіть істинні твердження:

1) в Java можно використовувати оператор **goto**;
2) в Java можно створювати методи, що не належать жодному классу;
3) Java підтримує множинне наслідування класів;
4) в Java не можна перегрузити оператор;
5) Java підтримує многопотоковість.

3. Дан код:

```
class Quest3 {
      public static void main(String s[ ])    {
      String args;
      System.out.print(args + s);
     } }
```

Результатом компіляции кода буде:

1) помилка компиляції: метод **main()** содержит неправильное имя параметра;
2) помилка компиляції: переменная **args** используется до инициализации;
3) помилка компиляції: несовпадение типов параметров при вызове метода **print()**;
4) компіляція безпомилок.

4. Які з наступних строк зкомпілюються без помилок?

1) float f = 7.0;
2) char c = "z";
3) byte b = 25 5;
4) boolean n = null;
5) int i = 325 65;
6) int j = 'ъ'.

5. Які варіанти запису оператора умовного переходу коректні?

1) if(i<j) { System.out.print("-1-"); }
2) if (i<j) then System.out.print("-2-");
3) if i<j { System.out.print("-3-"); }
4) if [i<j] System.out.print("-4-");
5) if (i<j) System.out.print("-5-");
6) if {i<j} then System.out.print("-6-");.

6. Які з наступних слів є ключовими чи резервованими словами в Java?

1) if;

2) then;
3) goto;
4) extend;
5) case.

7. Дан код:

```
public class Quest1 { static int i;
  public static void main(String[] args){ System.out.print(i);
  }
  }
```

В результаті при компіляції и запуску буде виведено:

1) ошибка компиляции: переменная i использована до инициализации;
2) null;
3) 1;
4) 0.

8. Які з ключових слів можуть бути використані при оголошенні конструктора?

1) private;
2) final;
3) native;
4) abstract;
5) protected.

9 Як слід викликати конструктор класа Quest3, щоб в результаті виконання коду була виведена на консоль строка "Конструктор".

```
public class Quest3 {
                    Quest3 (int i){ System.out.print("КоНСТруКТор");
}
      public static void main(String[] args){
        Quest3 s= new Quest3();
        //1
      }
  public Quest3()          {
        //2}
        {
        //3 }
        }
```

1) замість //1 написати `Quest3(1);`
2) замість //2 написати `Quest3(1);`
3) замість //3 написати new `Quest3(1);`
4) замість //3 написати `Quest3(1)` .

10. Дан код:

```
class Base {} class A extends Base {} public class Quest{
     public static void main(String[] args){
       Base b = new Base();
       A ob = (A) b;
      } }
```

Результатом компіляції і запуску буде:

1) компиляция и выполнение без ошибок;
2) ошибка во время компиляции;
3) ошибка во время выполнения.

11. Класи `A` і `Quest2` знаходяться в одному файлі. Що необхідно змінити в оголошенні класу, щоб воно було коректним?

```
    public class A{}
class Quest2 extends A, Object {}
```

1) необхідно прибрати специфікатор `public` перед `A`;

2) необхідно додати специфікатор `public` до `Quest2`;
3) прибрати після `extends` один з класів;
4) клас `Object` неможна вказувати явно.

12. Дан код:

```
class A {A(int i) {}} // 1 class B extends A
{}                          // 2
```

Якіз наступних тверджень вірні? (виберіть 2)

1) компилятор пытается создать по умолчанию конструктор для класса **A**;
2) компилятор пытается создать по умолчанию конструктор для класса **B**;
3) ошибка во время компиляции в строке 1;
4) ошибка во время компиляции в строке 2.

13. Які з фрагментів коду зкомпілюються без помилки?

1)

```
import java.util.*;
package First;
class My{/* тело класса*/}
```

2)

```
package mypack; import java.util.*;
public class First{/* тело класса*/}
```

3)

```
/*комментарий */
package first;
import java.util.*;
class First{/* тело класса*/}
```

13. Дан код:

```
abstract class QuestBase { static int i; abstract void show();
  }
public class Quest2 extends QuestBase {
  public static void main(String[] args){
  boolean[] a = new boolean[3];
   for(i = 0; i < a.length; i++)
   System.out.print(" " + a[i]);
   }
   }
```

В результаті при компіляції и запуске буде виведено:

1) false false false;
2) ошибка компиляции: массив а использован прежде, чем про- инициализирован;
3) ошибка компиляции: Quest2 должен быть объявлен как abstract;
4) ошибка времени выполнения: будет сгенерировано исключение IndexOutOfBounds;
5) true true true.

14. Які визначення інтерфейсу MyInterface є коректними ?

1)
```
interface MyInterface{
    public int result(int i){return(i++);}}
```
2)
```
interface MyInterface{
    int result(int i);}
```
3)
```
public interface MyInterface{
    public static int result(int i);}
```
4)
```
public interface MyInterface{
    public final static int i;
    {i=0;}
    public abstract int result(int i);}
```
5)
```
public interface MyInterface{
    public final static int i;
```

public abstract int result(int i);}

14. Дан код:

```
abstract class A {
 private int x = 2;
 public int show() {
return x;
}
}
interface B {int show();} class C {
      int x = 1;
   static class Nested extends A implements B {
    public int show() {
   return x++;}
}
}
```

Яке значення буде повернуте при виклику метода show()з классу Nested?

1) 1;
2) 2;
3) 3;
4) ошибка времени выполнения.
5) ошибка компиляции.

15. Що буде виведено в результаті компіляції та виконання наступного коду?

```
abstract class A {
      public int show() {
      return 1;}
}
class Quest2 {
      static int i = 2;
       static A ob1 = new A(){
      public int show() {return i++;}
      };
      static A ob2 = new A()   {
      public int show() {return ++i;}
      };
public static void main(String[] args) {
 System.out.print(ob1.show());
 System.out.print(ob2.show());
}
}
```

1) 34;
2) 24;
3) 33;
4) 23;
5) ошибка компиляции.

16. Які з оголошень коректні, якщо

```
class Owner{
      class Inner{
      } }
```

1) new Owner.Inner();
2) Owner.new Inner();
3) new Owner. new Inner();
4) new Owner().new Inner();
5) Owner.Inner();

6) Owner().Inner() .

17. Дан код:

```
public class Questl {
    public static void main(String[] args) {
        String str = new String("java");
        int i=1;
        char j=3;
        System.out.println(str.substring(i,j));
}
}
```

В результаті при компіляції і запуску буде виведено:

1) ja;
2) av;
3) ava;
4) jav;
5) ошибка компиляции: заданы некорректные параметры для метода substring().

18. Яку інструкцію слід використати, щоб визначити позицію букви v в рядку str = "Java"?

1) charAt(2,str);
2) str.charAt(2);
3) str.indexOf('v');
4) indexOf(str,'v');

19. Які з операцій коректні при оголошенні?

```
String s = new String("Java");
String t = new String();
String r = null;
```

1) r = s + t +r;
2) r = s + t +'r';
3) r = s & t &r;
4) r = s && t && r;

20. Дан код:

```
<applet code=MyApplet.class width=200 height=200> <param name=count value=5>
</applet>
```

Який код читає параметр count в змінну i?

1) int i = new Integer(getParameter("count")).intValue();
2) int i = getIntParameter("count");
3) int i = getParameter("count");
4) int i = new Integer( getIntParameter("count")).intValue();
5) int i = new Integer(getParameter("count"));.

21. В методі користувача show() було змінено колір фону апплета. Який метод сдід викликати, щоб це було візуалізовано?

1) setbgcolor();
2) draw();
3) start();
4) repaint();
5) setColor().

22. Які з класів наслідуються від класу Container?
   1) Window;
   2) List;
   3) Choice;
   4) Component;
   5) Panel;
   6) Applet;
   7) MenuComponent.

23. Яким чином в методі init() сервлета отримати параметр ініціалізації сервлета з іменем "URL"? (виберіть 2)
   1) Serv1etConfig.getInitParameter("URL");
   2) getServ1etConfig().getInitParameter("URL");
   3) this.getInitParameter("URL");
   4) HttpSerlet.getInitParameter("URL");
   5) ServletContext.getInitParameter("URL").

24. Який метод сервлета FirstServlet буде викликано при активізації посилання наступного HTML-документу?

```
<html><body>
  <a href="/FirstProject/FirstServ1ettest">OK!</a> </body><html>
```

Соответствующий сервлету тег <url-pattern> в файле web.xml имеет вид:

```
<url-pattern>/FirstServ1ettest</url-pattern>
```

   1) doGet();
   2) doGET();
   3) performTask();
   4) doPost();
   5) doPOST().

25. Контейнер викликає метод init() екземпляру сервлета...
   1) при кожному запиті до сервлету;
   2) при кожному запиті до сервлету, при якому створюється нова сесія;
   3) при кожному запиті до сервлету, при якому створюється новий поток;
   4) тільки один раз за життєвий цикл екземпляру;
   5) коли сервлет створюється вперше;
   6) якщо час життя сесії користувача, від якого надійшов запит, вичерпаний.

## *ПИТАННЯ ДЛЯ САМОКОНТРОЛЮ*

1. Області застосування Java.
2. Поняття про JDK, JVM. Java 2 (Java EE/SE/ME).
3. Установка JDK/ JRE, настройка параметрів середовища.
4. Eclipse як середовище розробки Java-додатків.
5. Основні етапи створення програми в середовищі Eclipse.
6. Установка Eclipse, перше знайомство з додатком, налагодження.
7. Поняття RTTI, інформація про клас; віртуальні методи;
8. Класи і інтерфейси; ідеологія Java.
9. Класи і пакети Java і їх співвідношення з елементами файлової системи.
10. Стандартні типи та об'єкти Java.
11. Посилання, покажчики і мова Java.
12. Об'єкти Java, цикл життя об'єктів;
13. Поняття про збір сміття;
14. Архіви Java. Створення простих демонстраційних додатків.
15. Класи, структура, область видимості, ієрархія класів.
16. Перевизначення методів класу.
17. Конструктори.
18. Інтерфейси, їх зміст і використання;
19. Інтерфейси і спадкування;
20. Перетворення типів з урахуванням класів і інтерфейсів в умовах поліморфізму.
21. Стандартні типи, їх об'єктні оболонки.
22. Масиви Java;
23. Стандартні типи Java - потоки введення-виведення , рядки , календар і ін..
24. Операції та їх пріоритети;.
25. Основні оператори Java, основні прийоми їх використання.
26. Два види об'єктів організації введення –виведення.
27. Файлове введення-виведення.
28. Консольне введення-виведення.
29. Види контейнерів Java. Основні прийоми використання контейнерів Java.
30. Використання шаблонів.
31. Роль сериалізації в Java. Стандартна процедура сериалізації.
32. Поняття про інтерфейс Cloneable.
33. Порівняння механізму з використанням винятків з традиційним механізмом обробки помилок.
34. Оброблювані і необроблювані виключення.
35. Стандартні виключення Java-технологій.
36. Оператори Java для підтримки винятків.
37. Компонентна модель JavaBeans.
38. Властивості, події, дескриптори компонентів.
39. Компоненти JavaBeans і обмін подіями в консольному додатку.

40. Сервлети Java.
41. Servlet API.
42. Інтерфейс сервлета.
43. Життєвий цикл сервлета.
44. Виклик сервлетів з HTML-сторінки.
45. ServletRequest інтерфейс.
46. ServletResponse інтерфейс.
47. Читання параметрів сервлета і формування відповіді клієнтові.
48. Читання параметрів ініціалізації.
49. Клас HttpServlet.
50. Робота з HTTP – запитами.
51. Сервер Jakarta Tomcat.
52. Поняття RTTI, інформація про клас; віртуальні методи;
53. Класи і інтерфейси; ідеологія Java.
54. Класи і пакети Java і їх співвідношення з елементами файлової системи.
55. Стандартні типи та об'єкти Java.
56. Посилання, покажчики і мова Java.
57. Об'єкти Java, цикл життя об'єктів;
58. Поняття про збір сміття;
59. Архіви Java. Створення простих демонстраційних додатків.
60. Класи, структура, область видимості, ієрархія класів.

## *СПИСОК ЛІТЕРАТУРИ*
## ОСНОВНА

1. Васильев А. Н. В. Java. Объектно-ориентированное программирование: Учебное пособие. — СПб.: Питер, 2011. — 400 с.

2. Ноутон П., Шилдт Г. Java 2: Наиболее полное руководство. СПб.: БХВ, 2008. 1072 с.

3. Хабибулин И. Самоучитель Java. СПб.: БХВ, 2008. 720 с.

4. Хорстманн К.С., Корнелл Г. Java 2. Библиотека профессионала. Т. 1, Основы. М.: Вильямс, 2014. 816 с.

5. Хорстманн К.С., Корнелл Г. Java 2. Библиотека профессионала. Т. 2, Тонкости программирования. М.: Вильямс, 2016. 992 с.

## ДОДАТКОВА

6. Монахов В. Язык программирования Java и среда NetBeans. СПб.: БХВ, 2012. 720 с.

7. Шилдт Г. Java 8: руководство для начинающих. М.: Вильямс, 2015. 720 с.

8. Шилдт Г. Java: методики программирования Шилдта. М.: Вильямс, 2008. 512 с.

9. Фишер Т. Р. Java. Карманный справочник. М.: Вильямс, 2008. 224 с.

10. Шилдт Г. Библиотека SWING для Java: руководство для начинающих. М.: Вильямс, 2007. 704 с.

11. Шилдт Г. Java 8.  Полное руководство. М.: Вильямс, 2016. 1376 с.

12. Эккель Б. Философия Java. СПб.: Питер., 2009. 640 с.

***ЗМІСТ***