1. Що таке Java-платформа?

*Програмна платформа* у загальному випадку включає набір програмних засобів, що забезпечують виконання інших програм. Програмні засоби, що складають програмну платформу, як правило, мають стандартний набір функцій, технічно по-різному реалізованих на різних апаратних платформах і в різних операційних системах. Java-платформу складають віртуальна машина Java і стандартні класи.

1. Чим відрізняється JDK і JRE?

* Java Runtime Environment (JRE) – середовище виконання Java-програм. Фізично це набір програмних засобів, що включають віртуальну машину Java і бібліотеку класів.
* Java Development Kit (JDK) – комплект розробника застосунків мовою Java, що включає в себе компілятор Java (javac), стандартні бібліотеки класів Java, приклади, документацію, різні утиліти і середовище виконання Java-програм (JRE); обидва комплекти розповсюджуються безкоштовно.

1. Що таке віртуальна машина Java?

*Віртуальна машина Java* (Java Virtual Machine) – це програма, яка інтерпретує і виконує байт-код, попередньо створений з вихідного тексту Java-програми компілятором Java.

1. Що таке байт-код?

Кожен клас компілюється в окремий файл, що містить двійковий код і має розширення .class. На відміну від С++, це не команди процесора, а байт-коди.

Скомпільована програма може бути виконана на комп'ютері, лише якщо на ньому встановлена так звана віртуальна машина Java (JVM, Java Virtual Machine) – спеціальна програма, яка здійснює інтерпретацію байт-кодів. Такий підхід дозволяє створювати програми, які можна переносити з одного операційного середовища (операційної системи, апаратної платформи) на інше без додаткової перекомпіляції. На відміну від C++, Java не передбачає статичного компонування коду у файл, який може бути виконано (.exe). Компонування (отримання байт-коду з різних файлів з розширенням .class або зі спеціальних архівів та інтерпретація коду) здійснюється динамічно.

1. Що таке Eclіpse?

*Eclipse* – це універсальна платформа, яка створена для побудови інтеґрованих середовищ розробки (IDE).

1. Як створити консольний застосунок у середовищі Eclipse?
2. Як запустити створену програму на виконання?

Для того, щоб виконати програму, слід створити конфігурацію виконання. У найпростішому випадку можна скористатися функцією Run | Run As | Java Application головного меню (або Run As | Java Application контекстного меню). У нижній частині вікна з'являється спеціальне консольне підвікно (Console).

1. Що таке "конфігурація часу виконання"?

Для того, щоб виконати програму, слід створити конфігурацію виконання. У найпростішому випадку можна скористатися функцією Run | Run As | Java Application головного меню (або Run As | Java Application контекстного меню). У нижній частині вікна з'являється спеціальне консольне підвікно (Console).

1. Які файли містять вихідний код на Java і які містять результат компіляції?

Кожен клас компілюється в окремий файл, що містить двійковий код і має розширення .class.

На відміну від C++, Java не передбачає статичного компонування коду у файл, який може бути виконано (.exe). Компонування (отримання байт-коду з різних файлів з розширенням .class або зі спеціальних архівів та інтерпретація коду) здійснюється динамічно.

1. Коли і як здійснюється компонування Java-програми?

На відміну від C++, Java не передбачає статичного компонування коду у файл, який може бути виконано (.exe). Компонування (отримання байт-коду з різних файлів з розширенням .class або зі спеціальних архівів та інтерпретація коду) здійснюється динамічно.

1. У чому полягають переваги та недоліки синтаксису мови Java (у порівнянні з С++)?

Синтаксис Java багато в чому схожий з C++. Разом з тим, Java має властивість крос-платформеності (транспортабельності), застосовує інтерпретатор байт-коду, використовує іншу *об'єктну модель* (аналогічну C#), тобто іншу сукупність об'єктно-орієнтованих концепцій і набору засобів опису класів і зв'язків між ними, а також механізмів створення об'єктів, та ін.

1. Яка різниця між ключовими і зарезервованими словами?

Набір ключових слів обмежений. Усі ключові слова є зарезервованими. Ключові слова не можна використовувати як ідентифікатори. У Java також існують зарезервовані слова, які не є ключовими. Їх взагалі не можна використовувати. Це **const** та **goto**.

1. Чим відрізняються коментарі Javadoc від інших видів коментарів?

* коментарі Javadoc (/\*\* \*/).

Спеціальний засіб javadoc.exe, що входить у JDK (Java Development Kit), використовує коментарі третього виду для автоматичної генерації документації. Для цього такі коментарі повинні бути відформатовані за стандартом Javadoc. Можна додавати коментарі для рівня класів і інтерфейсів, а також для рівня методів і полів. Коментарі Javadoc повинні бути розташовані перед відповідними фрагментами коду. Кожен коментар складається з опису і тегів. У коментарі javadoc можна включати теги форматування HTML. Не рекомендується включати теги, що забезпечують виведення жирним шрифтом, курсивом і т. д. Є спеціальний набір тегів Javadoc, таких як @author, @param, @return і т. д. Ці теги забезпечують стандартне представлення документації, що стосується класів і методів. Наприклад, такі коментарі перед методом

1. Як створюється вбудована документація?
2. Чим відрізняються примітивні типи даних Java від аналогічних типів C++?

У багатьох інших мовах (наприклад, у С та у С++) формат і розміри примітивних типів даних залежать від платформи (DOS, Win 32 і т. д.). У Java розміри і формат стандартизовані і не залежать від платформи.

1. Для чого використовують керуючі послідовності?

'\n' – новий рядок,

'\t' – горизонтальна табуляція,

'\r' – переведення на початок рядку,

'\'' – одиночні лапки (апостроф),

'\"' – подвійні лапки,

'\\' – зворотна коса риска (backslash).

1. Як здійснюється приведення типів?

**int** i = 10;

**float** f = i; // Таке перетворення допускається

**long** l = f; // Помилка!

**long** l1 = (**long**) f; // Явне приведення типів

1. У чому полягають особливості типу boolean?

Булевим змінним можна присвоювати тільки константи**true** та **false**. Змінні типу **boolean** не можна неявно чи явно приводити до інших типів і навпаки.

1. Як привести значення вираз типу boolean до цілого типу?

Змінні типу **boolean** не можна неявно чи явно приводити до інших типів і навпаки.

1. Чим відрізняється префіксна та постфіксна форма операцій інкременту та декременту?

 Префіксна форма забезпечує збільшення змінної до того, як значення операції буде використано, а постфіксна – після. *Операція декременту* -- забезпечує зменшення змінної на одиницю і правила її використання аналогічні.

1. Чи можна застосовувати операцію & замість && для операндів типу boolean?

так

1. Чи завжди потрібна мітка при використанні break?
2. Чи можна розмістити декілька пакетів у одному вихідному файлі?

*Пакет* – це набір взаємозалежних класів та інших елементів коду, які здійснюють взаємодію один з одним через загальний доступ до функцій і даних і через загальний простір імен

Пакети можуть вкладатися один в інший. Кожен клас Java знаходиться в будь-якому пакеті

1. Чим відрізняються формальні та фактичні параметри функцій?

Параметри (аргументи) функції, що вказуються в списку у визначенні функції, називаються *формальними*. Параметри, що вказуються під час виклику функції, називаються *фактичними*. Під час виклику функції виділяється пам'ять під її формальні параметри, потім кожному формальному параметру присвоюється значення фактичного параметра.

1. Чи можна використовувати інструкцію return у тілі функції з типом результату void?

**void** hello() {

System.out.println("Hello!");

}

У цьому випадку в тілі функції **return** може бути відсутнім. Якщо інструкція **return** присутня, то після неї не повинно бути ні якого виразу. Таку функцію можна викликати тільки окремою інструкцією.

1. У чому полягає перевантаження імен функцій?

Якщо списки формальних параметрів двох функцій розрізняються числом чи параметрів їхніми типами, то ці дві функції вважаються перевантаженням однієї функції і повинні мати різні визначення.

Перевантажені функції використовуються в тих випадках, якщо кілька функцій виконує схожі дії над об'єктами різних типів і зручно дати однакові імена всім цим функціям:

**static int** max(**int** a, **int** b) { } // Вибір максимального з двох цілих чисел

**static int** max(**double** x, **double** y, **double** z) { } // Вибір максимального з трьох дійсних чисел

1. Чи можна звертатися до імен з іншого пакета, не використовуючи операції import?

Якщо класи знаходяться в інших пакетах (не у java.lang), для виклику цих функцій слід або застосовувати префікс – ім'я\_пакету.ім'я\_класу. Якщо необхідні класи вкладених пакетів, префікс буде ще більш складним. Для того, щоб запобігти використанню повних імен, застосовується конструкція import

1. У чому є переваги та недоліки статичного імпорту?

Додаткова форма імпорту, яка з'явилася у версії Java 5, дозволяє імпортувати тільки статичні елементи зазначеного класу. Наприклад, у програмах, які реалізують математичні обчислення доцільно додавати статичний імпорт елементів класу Math:

**import** **static** java.lang.Math.\*;

Тепер статичні елементи можна використовувати без додаткового кваліфікатору, наприклад:

**double** d = sin(1);

1. Як виконати скомпільовану програму, не використовуючи середовище Eclipse?

Припустимо, проект First створений у робочому просторі, зв'язаним з текою E:\eclipse\workspace, а функція main() реалізована в класі TestClass пакету first. Для запуску програми в командному рядку варто набрати

java -cp E:\eclipse\workspace\First first.TestClass