1. **Назвіть основні складові платформи Java SE.**
2. **Чим відрізняються потоки байтів від потоків символів за областю застосування?**

чем?

1. **Які класи забезпечують роботу з текстовими файлами і бінарними файлами?**

Потоки, призначені для роботи з текстовою інформацією, мають назву потоків символів. Імена класів таких потоків закінчуються відповідно словами "...Reader" і "...Writer". FileReader та FileWriter.

Для роботи з нетекстовими (бінарними) файлами використовують потоки, імена яких замість "Reader" або "Writer" містять "Stream", наприклад InputStream, FileInputStream, OutputStream, FileOutputStream тощо. Такі потоки мають назву потоків байтів

1. **У чому сенс явного закриття файлів?**

Чтобы освободить ресурсы.

Вот практический пример, почему это "хороший стиль". Вы пишете некую программу и не закрываете ресурсы, понадеявшись на ОС. Но затем Ваш код из самостоятельной единицы превращается в часть кода другой, более мощной программы. Но Ваш код не освобождает ресурсы и в результате эти ресурсы "висят" невостребованными. Более того, Ваш код выполняется много раз и каждый раз открывает ресурс и не закрывает его. Такое поведение по меньшей мере приводит к напрасному расходу этих ресурсов, а в более глобальной перспективе замедляет работу не только программы, но и компьютера, на котором она выполняется.

1. **Чи можна одночасно відкрити кілька потоків введення / виведення?**

**+**

1. **Яким чином можна забезпечити автоматичне закриття потоків?**

Крім того, доцільно скористатися конструкцією try () { } Java 7 для автоматичного закриття потоку:

...

try (Scanner scanner = new Scanner(new FileReader("data.txt"))) {  
 try (PrintWriter pw = new PrintWriter("results.txt")) {

…

}

}

...

1. **У чому переваги використання класу RandomAccessFile?**

Об'єкт цього класу дозволяє вільно пересуватися всередині файлу в прямому і зворотному напрямку. Основною перевагою класу RandomAccessFile є можливість читати і записувати дані в довільне місце файлу.

1. **Для чого використовують файли даних DataOutputStream і DataInputStream? Які у них переваги і недоліки?**

Так звані потоки даних (data streams) підтримують бінарне введення / виведення значень примітивних типів даних (boolean, char, byte, short, int, long, float і double), а також значень типу String. Усі потоки даних реалізують інтерфейси DataInput або DataOutput. Для більшості задач достатньо стандартних реалізацій цих інтерфейсів - DataInputStream і DataOutputStream. Дані у файлі зберігаються в такому вигляді, в якому вони представлені в оперативній пам'яті

1. **Що таке серіалізація і для чого вона використовується?**

Сериализация это процесс сохранения состояния объекта в последовательность байт; десериализация это процесс восстановления объекта, из этих байт. Java Serialization API предоставляет стандартный механизм для создания сериализуемых объектов

Серіалізація та десеріалізація можуть бути використані замість файлового введення та виведення.

1. **У чому є переваги й недоліки серіалізації?**

Головним недоліком бінарної серіалізації є необхідність роботи з двійковими (нетекстовими) файлами.

Персистентность объекта это способность обекта жить или по-другому - "пережить" выполнение программы. Это значит, что любой объект, который был создан во времени выполнения, уничтожается "мусорщиком" JVM, всякий раз, когда данный объект далее перестает использоваться. Но в случае реализации API персистентности, данные объекты не будут уничтожаться "мусорщиком" JVM, вместо чего им будет позволено "жить", что также дает возможность доступа к ним при следующем запуске приложения. Другими словами, персистенция означает существование времени жизни объекта, независимо от времени жизни приложения, которое запущено.  
  
Один из способов реализации персистентности это хранение объектов где-нибудь во внешнем файле или в базе данных, а затем восстановление их в более позднее время, используя данные файлы или базу данных как источники. Здесь сериализация и вступает в игру. Любой неперсистентный объект существует так долго, как долго работает JVM.  
  
Сериализованные объекты – это просто объекты, преобразованные в потоки, которые затем сохраняются во внешний файл или передаются через сеть для хранения и восстановления.

1. **Які функції слід визначити для реалізації інтерфейсу java.io.Serializable?**

Для того, щоб об'єкти певного класу можна було серіалізувати, клас повинен реалізовувати інтерфейс java.io.Serializable. Цей інтерфейс не визначає жодного метода, його наявність лише вказує, що об'єкти цього класу можна серіалізувати. Однак гарантована серіалізація і десеріалізація вимагає наявності в таких класах спеціального статичного поля serialVersionUID, яке забезпечує унікальність класу.

1. **Для чого використовують модифікатор transient?**

Деякі поля класу, значення яких не впливають на стан об'єкта, можна описати з модифікатором transient. Такі поля не будуть збережені у потоці під час серіалізації та не будуть відтворені під час десеріалізації.

1. **Як в Java здійснюється робота з архівами?**

Пакет java.util.zip надає можливості роботи зі стандартними файлами ZIP і GZIP форматів.  
  
Для запису в архів застосовують клас ZipOutputStream. За допомогою функції setMethod() цього класу можна визначити метод архівації - ZipOutputStream.DEFLATED (з компресією) або ZipOutputStream.STORED (без компресії). Метод setLevel() визначає рівень компресії (вд 0 до 9, за умовчанням Deflater.DEFAULT\_COMPRESSION, зазвичай, максимальна компресія). Метод setComment() дозволяє додати коментар до архіву.

Для того, щоб прочитати дані з архіву, необхідно скористатися класом ZipInputStream. У кожному такому архіві завжди потрібно переглядати окремі записи (entries). Метод getNextEntry() повертає об'єкт типу ZipEntry. Метод read() класу ZipInputStream повертає -1 наприкінці поточного запису (а не тільки в кінці Zip-файлу). Далі викликається метод closeEntry() для отримання можливості переходу до зчитування наступного запису. В наступному прикладі здійснюється читання запсу ZipCreator.java з раніше створеного архіву та виведення його вмісту в консольне вікно:

1. **Чи можна створити архів з кількома файлами всередині?**

putNextEntry(zipEntry)

1. **Для яких цілей використовуються XML-документи?**

Розширювана мова розмічування XML (eXtensible Markup Language) - це незалежний від платформи метод структурування інформації. Оскільки XML відокремлює зміст документу від його структури, його успішно використовують для обміну інформацією. Наприклад, XML можна використовувати для передачі даних між програмою та базами даних, або між базами даних, що мають різні формати.

1. **Які обмеження накладаються на структуру XML-документу, синтаксис і розташування тегів?**

Файли формату XML - це завжди текстові файли. Синтаксис мови XML багато в чому схожий на синтаксис мови HTML, яка застосовується для розмічування текстів, що публікуються в Internet. Мова XML також може бути безпосередньо застосована для розмітки текстів.

Елементи розділені тегами. Початкові теги починаються з символу < з розміщеним далі ім'ям елементу. Кінцеві теги починаються з символів </, за якими слід ім'я елементу. Початковий і кінцевий теги закінчуються символом >

На відміну від HTML, XML дозволяє використовувати необмежений набір пар тегів, кожна з яких представляє не те, як включені в неї дані повинні виглядати, а те, що вони означають. XML дозволяє створювати свій набір тегів для кожного класу документів. Таким чином, точніше назвати його не мовою, а метамовою.

Теги можуть містити в собі атрибути - додаткову інформацію про елементи, яка міститься всередині кутових дужок. Значення атрибутів обов'язково брати у лапки.

Важливим правилом формування XML є обов'язковість вживання кінцевих тегів. Крім того не можна плутати порядок кінцевих тегів.

У XML-документі можна вживати коментарі, синтаксис яких збігається з синтаксисом коментарів HTML-документів:  
  
<!-- Це коментар -->

1. **Чим відрізняються технології SAX і DOM?**

Існує два стандартних підходи до роботи з XML-документами в програмах:  
  
(Simple API for XML, SAX), що працює на потоці даних, обробляючи його, коли виникають пов'язані з різними тегами події;  
об'єктна модель документу (Document Object Model, DOM), що дозволяє створити в пам'яті колекцію пов'язаних з тегами об'єктів, організованих в ієрархію.

DOM парсер сначала загружает весь исходный документ в java объект, с которым затем можно работать; SAX парсер использует событийную модель разбора документа, парсинг происходит "на лету" и загружать в память сразу весь XML нет необходимости.

1. **Яким чином здійснюється читання і запис XML-документів?**

Програми розпізнавання XML-документів - так звані XML-парсери - здійснюють розбір документа до знаходження першої помилки, на відміну від HTML-парсерів, вбудованих в браузер. Браузери намагаються відобразити документ, навіть, якщо код містить помилки.

Базований на подіях підхід не дозволяє розробнику змінювати дані в вихідному документі. В разі необхідності коригування частини даних документ треба повністю оновити. На відміну від нього DOM забезпечує API, який дозволяє розробнику додавати або видаляти вузли в будь-якій точці дерева в застосунку.

Обидва підходи використовують поняття парсеру. Парсер (parser) - це програмний застосунок, призначений для того, щоб аналізувати документ шляхом розділення його на лексеми (tokens). Парсер може ініціювати події (як у SAX), або будувати в пам'яті дерево даних.

Для реалізації стандартних підходів до роботи з XML в Java SE використовують засоби Java API for XML Processing (JAXP, інтерфейс програмування застосунків Java для роботи з XML). JAXP надає засоби валідації і розбору XML-документів. Для реалізації об'єктної моделі документа JAXP включає програмний інтерфейс DOM, SAX реалізований однойменною програмним інтерфейсом. На додаток до них надано програмний інтерфейс Streaming API for XML (StAX, потоковий API для XML), а також засоби XSLT (XML Stylesheet Language Transformations, мова перетворення XML-документів).

1. **Що таке XSLT?**

XML Stylesheet Language Transformations, мова перетворення XML-документів

Преобразования расширяемого языка таблиц стилей (XSLT) предоставляют способ для автоматического перевода XML-данных из одной формы в другую. Целевая форма - это обычно другой XML-документ, но не обязательно; вы можете преобразовать XML практически во что угодно, просто создав таблицу стилей XSLT и обработав данные. Если вы хотите изменить результаты, вы просто меняете таблицу стилей и обрабатываете XML заново. Здесь есть дополнительное преимущество, расширяющее возможности непрограммистов, например, дизайнеров, которые могут изменять таблицу стилей и влиять на результаты.

Простейший способ выполнить XML-преобразование - это добавить указание на таблицу стилей в XML и отобразить его в браузере

1. **Чим відрізняється валідний (valid) та правильно оформлений (well-formed) XML-документ?**

XML-документ, який є правильно оформленим, посилається на граматичні правила та повністю їм відповідає, має назву валідного (valid) документу.

XML-документ, який відповідає всім синтаксичним правилам XML, вважається правильно оформленим документом (коректним документом, well-formed document).

Проте, XML-документ, що містить DTD, або вказує на нього, вважається "правильним" (valid).

1. **Чим відрізняються шаблони документу і схеми документу?**

Структуровані дані, які можуть бути представленими у формі XML-файлу, потребують додаткової інформації. Найбільш розповсюдженими є два основних формати представлення такої інформації - Визначення шаблону документу (DTD) та Схема документу (XSD).

DTD (Document Template Definition) - набір правил, що дозволяють однозначно визначити структуру певного класу XML-документів.DTD не є XML-документом. DTD дуже простий, але не описує типів елементів. Директиви DTD можуть бути присутніми як у заголовку самого XML-документу (internal DTD), так і в іншому файлі (external DTD). Наявність DTD не є обов'язковою. Проте, XML-документ, що містить DTD, або вказує на нього, вважається "правильним" (valid).

XML Schema - це альтернативний DTD спосіб завдання структури документу. Схема зручніше, ніж DTD тим, що опис структури документу виконано на самому XML, у той час як DTD використовує директиви, синтаксис яких не має нічого спільного з синтаксисом XML. Крім того, XML схема своїми можливостями істотно перевершує DTD. Наприклад, у схемі можна вказувати типи тегів та атрибутів, визначати обмеження тощо.

1. **Чи є шаблон документу XML-документом?**

DTD не є XML-документом.

1. **Чи є схема документу XML-документом?**

**+**

1. **Для чого в XML-документах необхідні простори імен?**

Для того щоб запобігти конфліктам імен тегів, у XML можна створювати так звані простори імен. Простір імен визначає префікс, пов'язаний з певною схемою документу та додається до тегів. Власний простір імен визначається за допомогою конструкції на кшталт такої:  
  
<root xmlns:pref="http://www.someaddress.org/">  
У цьому прикладі root - кореневий тег XML-документу, pref - префікс, який визначає простір імен, "http://www.someaddress.org/" - якась адреса, наприклад, доменне ім'я автора схеми. Програми, які обробляють XML-документи, ніколи не перевіряють цю адресу. Вона необхідна лише для забезпечення унікальності простору імен.  
  
Безпосередньо схема використовує простір імен xs.

Створення файлу схеми слід починати зі стандартної конструкції:  
  
<?xml version="1.0" encoding="UTF-8"?>  
<xs:schema xmlns:xs="http://www.w3.org/2001/XMLSchema">  
  
</xs:schema>

1. **Що таке маршалізація і демаршалізація?**

Засіб зв'язування даних XML містить компілятор схеми, що транслює схему в набір специфічних для схеми класів з відповідними методами доступу і зміни (тобто get і set). Він також містить механізм маршалізації (запису структурованих даних в XML-документ), підтримує демаршалізацію XML документів у відповідну структуру взаємозалежних екземплярів. Автоматично створеною структурою даних можна користуватися без ручного розміщення даних у списках або масивах.

Маршалинг (от англ. marshal — упорядочивать) в информатике — процесс преобразования информации (данных, двоичного представления объекта), хранящейся в оперативной памяти, в формат, пригодный для хранения или передачи. Процесс похож на сериализацию (отличия см. ниже). Обычно применяется тогда, когда информацию (данные, объекты) необходимо передавать между различными частями одной программы или от одной программы к другой.

1. **У чому переваги технології зв'язування даних?**

Технологія зв'язування даних найбільш часто застосовується для генерації класів за наявною схемою. Спочатку в теці проекту треба створити файл схеми (\*.xsd).

1. **Які є стандартні й нестандартні технології зв'язування даних?**

Традиційно першою технологією зв'язування даних була технологія Castor. Пізніше був стандартизований API JAXB (Java Architecture for XML Binding). Версія 2 специфікації JAXB припускає як генерацію класів за схемою, так і генерацію схеми за існуючою структурою класів.

1. **Які налаштування проекту слід встановити для використання засобів Dali Java Persistence Tools?**

Для підтримки стандарту технологій API JAXB в середовищі Eclipse Oxygen повинні бути встановлені засоби Dali Java Persistence Tools (JAXB Support). Якщо в Eclipse не встановлені необхідні програмні засоби, їх можна додати за допомогою головного меню Eclipse Help | Install New Software, далі в рядку Work with: обираємо Oxygen - http://download.eclipse.org/releases/oxygen. Далі знаходимо в списку потрібні програмні засоби і натискаємо Next, на сторінці Install Details знову натискаємо Next, далі слід погодитися з умовами ліцензії й натиснути Finish. Після завантаження нового програмного забезпечення слід перезавантажити Eclipse.

1. **Які класи відповідають специфікації Java Beans?**

Java Bean - це клас, що задовольняє наступним вимогам:  
  
клас відкритий (public)  
відсутні відкриті дані (відкритими можуть бути тільки методи)  
клас повинен містити усталений конструктор (конструктор без аргументів)  
клас повинен реалізовувати інтерфейс java.io.Serializable  
пара методів з іменами setNnn() і getNnn() утворюють властивість з іменем nnn і відповідним типом. Для властивостей типу boolean використовують "is" замість "get".

1. **Які є недоліки й переваги XML-серіалізації?**

Головним недоліком описаної раніше бінарної серіалізації є необхідність роботи з двійковими (нетекстовими) файлами. Зазвичай такі файли використовують не для довгострокового зберігання даних, а для одноразового зберігання і відновлення об'єктів. Безумовно, більш зручною й керованою є серіалізація в текстовий файл, зокрема, в XML-документ. Існує декілька підходів до серіалізації й десеріалізації, побудованої на XML. Найбільш простим є підхід, заснований на використанні класів java.beans.XMLEncoder і java.beans.XMLDecoder. Найбільш природне застосування цих класів - зберігання та відтворення елементів графічного інтерфейсу. Але можна також зберігати об'єкти інших класів, які відповідають специфікації Java Beans

1. **Як визначити поняття "файлова система"?**

Фа́йловая систе́ма (англ. file system) — порядок, определяющий способ организации, хранения и именования данных на носителях информации в компьютерах, а также в другом электронном оборудовании: цифровых фотоаппаратах, мобильных телефонах и т. п. Файловая система определяет формат содержимого и способ физического хранения информации, которую принято группировать в виде файлов. Конкретная файловая система определяет размер имен файлов (и каталогов), максимальный возможный размер файла и раздела, набор атрибутов файла. Некоторые файловые системы предоставляют сервисные возможности, например, разграничение доступа или шифрование файлов.

1. **Які можна назвати типові функції для роботи з файловою системою?**

Клас File містить методи для отримання списку файлів визначеної теки (list(), listFiles()), отримання та модифікації атрибутів файлів (setLastModified(), setReadOnly(), isHidden(), isDirectory() тощо), створення нового файлу (createNewFile(), createTempFile()), створення тек (mkdir()), видалення файлів та тек (delete()) та багато інших

1. **Які засоби надає Java для роботи з файловою системою?**

Функція list() без параметрів дозволяє отримати масив рядків - усіх файлів та підкаталогів теки, визначеної під час створення об'єкта типу File. Виводяться відносні імена файлів (без шляхів). У наступному прикладі ми отримуємо список файлів та підкаталогів теки, ім'я якої вводиться з клавіатури:

На відміну від list(), функція listFiles() повертає масив об'єктів типу File. Це надає додаткові можливості - отримання імен файлів з повним шляхом, перевірки значень атрибутів файлів, окрему роботу з теками тощо.

Для визначення маски-фільтру необхідно створювати об'єкт класу, який реалізує інтерфейс FilenameFilter.

1. **Як отримати атрибути файлу за допомогою засобів класу java.io.File?**

Клас File містить методи для отримання списку файлів визначеної теки (list(), listFiles()), отримання та модифікації атрибутів файлів (setLastModified(), setReadOnly(), isHidden(), isDirectory() тощо

1. **Чим відрізняються функції list() і listFiles()?**

Функція list() без параметрів дозволяє отримати масив рядків - усіх файлів та підкаталогів теки, визначеної під час створення об'єкта типу File. Виводяться відносні імена файлів (без шляхів)

На відміну від list(), функція listFiles() повертає масив об'єктів типу File. Це надає додаткові можливості - отримання імен файлів з повним шляхом, перевірки значень атрибутів файлів, окрему роботу з теками тощо.

1. **Як здійснити копіювання файлів?**

private static void copyFileUsingJava7Files(File source, File dest) throws IOException {  
 Files.copy(source.toPath(), dest.toPath());  
}