1. **Що таке патерн проектування?**

Патерн проектування (design pattern, проектний зразок, шаблон) - це опис взаємодії об'єктів і класів, адаптований для вирішення певної задачі в конкретному контексті.

1. **Опишіть складові частини патерну MVC.**

Раніше згадувався засадничий патерн проектування Model-View-Controller (MVC, «модель-вигляд-контролер»). Він був вперше застосований в системі Smalltalk-80. MVC може навіть розглядатися не як шаблон, а як основоположна концепція проектування, згідно з якою модель даних програми (бізнес-логіка), інтерфейс користувача (візуальне уявлення) і взаємодія з користувачем відокремлені один від одного.  
  
Модель надає дані і методи роботи з даними і не містить інформації про візуальному поданні цих даних.  
  
Вигляд (представлення) відповідає за візуальне представлення інформації. Наприклад, це графічний інтерфейс користувача, вікно з графічними елементами.  
  
Контролер забезпечує зв'язок між користувачем, моделлю і представленням.  
  
Такий розподіл дозволяє будувати модель незалежно від візуального представлення, а також створювати кілька різних представлень для однієї моделі.

пакет model міститиме класи, які описують предметну область (Census, XMLCountry тощо);  
пакет view з FXML-документами;  
пакет controller з класом-контролером.

1. **Коли й ким вперше були систематично описані патерни проектування?**

Вперше патерни проектування були систематично викладені в книзі "Патерни проектування: Елементи повторно використовуваного об'єктно-орієнтованого програмного забезпечення" ("Design Patterns: Elements of Reusable Object-Oriented Software", "Приемы объектно-ориентированного проектирования. Паттерны проектирования") авторів Е. Гамма Р. Хелм Р. Джонсон Дж. Вліссідес (http://www.uml.org.cn/c++/pdf/DesignPatterns.pdf (англ) http://www.sugardas.lt/~p2d/books/Priemioop.pdf, рос.). Ця книга вийшла англійською мовою в 1995 р у видавництві Addison Wesley Longman, Inc. Подальший розвиток патернів відображено в книзі "Застосування UML і шаблонів проектування" автора Крега Лармана.

1. **Визначення яких елементів включає ідентифікація патерну?**

Ідентифікація патерну передбачає визначення таких елементів:  
  
ім'я; присвоювання шаблонами імен дозволяє проектувати на більш високому рівні абстракції;  
задача; опис того, коли слід застосовувати;  
рішення; опис елементів проектного рішення, відносин між ними, функцій кожного елемента;  
результати; наслідки застосування шаблону і можливі компроміси

1. **Які специфікації включає стандартний опис патерну?**

Стандартний опис патерну включає визначення таких основних специфікацій:

найменування та класифікація проектного зразка;

призначення;

"відомий також як";

мотивація;

придатність;

структура;

учасники;

відносини;

результати;

реалізація;

приклад коду;

відомі застосування;

споріднені шаблони проектних рішень.

1. **На яких механізмах базуються патерни?**

Шаблони проектних рішень базуються на механізмах повторного використання. До них відносяться:  
  
успадкування та композиція;  
делегування;  
параметризовані типи.

1. **Наведіть класифікацію патернів проектування.**

порождающие

структурные

поведения

обязательств

1. **У чому полягає ідея та реалізація патерну Factory Method?**

Factory Method (Фабричный метод) - твірний патерн, який надає підкласам інтерфейс для створення екземплярів деякого класу. У момент створення спадкоємці можуть визначити, який клас створювати. Фабрика делегує створення об'єктів спадкоємцям батьківського класу. Це дозволяє використовувати в коді програми не специфічні класи, а маніпулювати абстрактними об'єктами на більш високому рівні. Також відомий під назвою віртуальний конструктор (Virtual Constructor).

Інший підхід базується на використанні віртуальних методів. Припустимо, є абстрактний клас AbstractNumber, який оголошує абстракті функції знаходження суми двох чисел і абстрактний фабричний метод, а також реалізує метод знаходження суми елементів масиву чисел. Абстрактний фабричний метод застосовується для створення початкового значення суми:

1. **Як реалізувати патерн Factory Method мовою Java?**

создать класс фабрику (ShapeFactory), интерфейс (Shape), классы (e.g. Circle, Rectangular), которые имплементируют Shape. В классе ShapeFactory создать getter, который в зависимости от входных параметров возвращает Объект типа Shape либо Circle, либо Rectangular

есть абстрактный класс AbstractNumber, в котором определен абстрактный метод getIntance, возвращающий AbstractNumber. Создаются производные от AbstractNumber классы (e. g. IntegerNumber) в котором перегружен метод getInstance, использующийся в функциях родительского класса.

1. **У чому призначення патерну Abstract Factory?**

Подає інтерфейс для утворення родин взаємозв'язаних або взаємозалежних об'єктів, не специфікуючи їхніх конкретних класів.

Слід використовувати шаблон Абстрактна фабрика коли:  
  
система не повинна залежати від того, як утворюються, компонуються та представляються вхідні до неї об'єкти;  
вхідні до родини взаємозв'язані об'єкти повинні використовуватися разом і необхідно забезпечити виконання цього обмеження;  
система повинна конфігуруватися однією з родин складаючих її об'єктів;  
треба подати бібліотеку об'єктів, розкриваючи тільки їхні інтерфейси, але не реалізацію.

1. **Для чого використовують патерн Lazy Initialization?**

Патерн Lazy initialization передбачає створення об'єкта безпосередньо перед тим, як цей об'єкт буде вперше використано. Таким чином, ініціалізація виконується "на вимогу", а не завчасно.

1. **Як реалізувати патерн Lazy Initialization мовою Java?**

Типу присваивается значение null, а сам объект создается в первом методе, который к нему обращается

1. **Коли використовують патерн Singleton?**

Патерн Singleton (Одинак) застосовують в тому випадку, коли який-небудь клас може мати тільки один екземпляр (або не мати жодного). Це може бути будь-якої великий об'єкт, який не повинен тиражуватися. Наприклад, об'єкт може бути пов'язаний з файлом, в якому фіксується налагоджувальна інформація тощо.  
  
Використання статичних даних та методів замість об'єкта-одинака не завжди допустимо, оскільки відповідні елементи даних створюються автоматично під час першого звернення до класу, а нам потрібне рішення, яке дозволяє контролювати створення об'єкта-одинака.

Рішення полягає в тому, щоб сам клас контролював свою «унікальність», забороняючи створення нових екземплярів, та сам забезпечував єдину точку доступу. Це є призначенням шаблону Одинак.

1. **Як реалізувати патерн Singleton мовою Java?**

Создается класс одиночка (напр. Singlton), в нем есть объект с именем instance типа static final Singlton и метод getInstance, возвращающий этот объект. final обеспечивает то, что других объектов этого класса создаваться не будет. Конструктору класса Singlton назначается модификатор private

1. **У чому призначення патерну Adapter?**
2. **Як мовою Java реалізувати патерн Adapter?**
3. **Коли і для чого застосовують патерн Facade?**

Патерн Facade шаблон проектування , призначений для об'єднання групи підсистем під один уніфікований інтерфейс, надаючи доступ до них через одну точку входу  
  
Оскільки клас-фасад може бути застосований для створення лише одного об'єкта, такий клас доцільно реалізовувати з використанням патерну Singleton

1. **Як зв'язаний патерн Facade з патерном MVC?**
2. **У чому є ідея і реалізація патерну Observer?**
3. **Наведіть приклади використання патерну Observer у JavaFX.**
4. **Як слід застосовувати патерни для розв'язання задач проектування програмного забезпечення?**
5. **Наведіть приклади використання композиції та поліморфізму в патернах проектування.**