Question 1

Question Type: MultipleChoice

You are building an tool that imports products from an ERP. There are 20 columns of additional information that are associated with each product. This extra information must also be associated with an update time to know when to refresh the dat

a. Keeping maintainability in mind, how do you build this into Magento?

Options:

A- Utilize an extension attribute.

B- Override the Product model and add the fields.

C- Create a separate model and build code to associate the two record types.

D- Create 20 EAV attributes and check their updated_at column.

Answer:

A

Question 2

Question Type: **MultipleChoice**

You are reviewing a module to some special functionality to the Magento 2 application, You see directory /CustomerData, What task you think in this directory contain script for this modules?

Options:

A- Contains section files that works with the data stored on the client side.

B- Contains aggregated functionality.

C- Contains localization files.

D- Contains view files, including static view files, design templates, email templates, and layout files.

Answer:

A

Question 3

Question Type: **MultipleChoice**

You are building CLI that use the console to create a customer account with our custom command as like php bin/magento customer:user:create --customer-firstname="Mahin" --customer-lastname="Rahman" --customer-email="mahin@example.com" --customer-password="mahin@123" --website="1"

using: protected function configure() { $this->setName('customer:user:create') ->setDescription('Create new customer') ->setDefinition($this->getOptionsList()); } protected function getOptionsList(){ return [

-----------------]; } Which below Option are not required in blank? (Choose 2)

Options:

A- new InputOption(Customer::KEY_SENDEMAIL, 0, InputOption::VALUE_OPTIONAL, '(1/0) Send email? (default 0)')

B- new InputOption(Customer::KEY_STORE, null, InputOption::VALUE_REQUIRED, '(Required) Store ID'),

C- new InputOption(Customer::KEY_PASSWORD, null, InputOption::VALUE_REQUIRED, '(Required) Customer password'),

D- new InputOption(Customer::KEY_EMAIL, null, InputOption::VALUE_REQUIRED, '(Required) Customer email'),

E- new InputOption(Customer::KEY_LASTNAME, null, InputOption::VALUE_REQUIRED, '(Required) Customer last name'),

F- new InputOption(Customer::KEY_FIRSTNAME, null, InputOption::VALUE_REQUIRED, '(Required) Customer first name'),

G- new InputOption(Customer::KEY_WEBSITE, null, InputOption::VALUE_REQUIRED, '(Required) Website ID'),

Answer:

A, B

Question 4

Question Type: **MultipleChoice**

You need to render a product attribute's raw value as a variable in a script tag. This value will be used to initialize an application on the frontend. How do you render this value?

Options:

A- <?= $block->escapeJs($value) ?>

B- <?= $block->escapeJsVar($value) ?>

C- <?= $block->renderJs($value) ?>

D- <?= $block->escapeHtml($value) ?>

Answer:

A

Question 5

Question Type: **MultipleChoice**

As you are scanning folder in the vendor/module-catalog directory, you see a directory that is named Ui. What is this folder's purpose?

Options:

A- It contains UI component data providers and component information.

B- It contains templates, CSS and JS pertinent to the module.

C- It contains the block PHP files that render HTML onto the frontend.

D- It is not a normal folder and further investigation is necessary to determine the purpose.

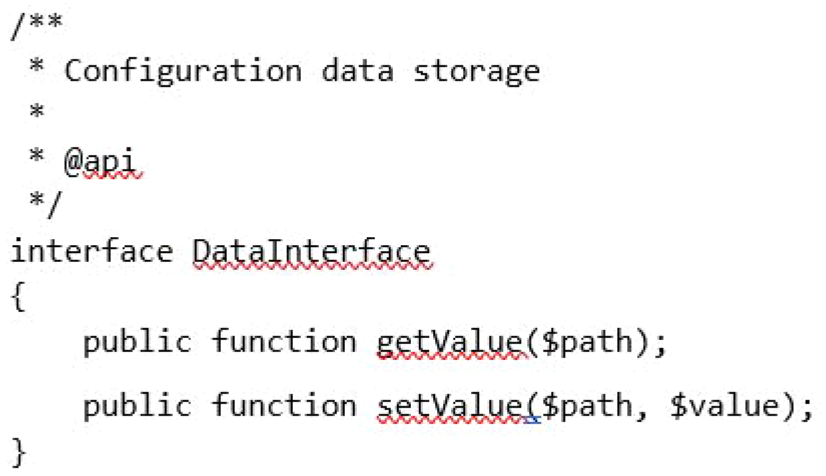
Answer:

A

Question 6

Question Type: **MultipleChoice**

In a custom module you implement the interface \Magento\Framework\App\Config\DataInterface.



```
/**
 * Configuration data storage
 *
 * @api
 */
interface DataInterface
{
    public function getValue($path);

    public function setValue($path, $value);
}
```

What version constraint for magento/framework do you add to your module's composer.json file?

Options:

A- major

B- minor

C- patch

D- stable

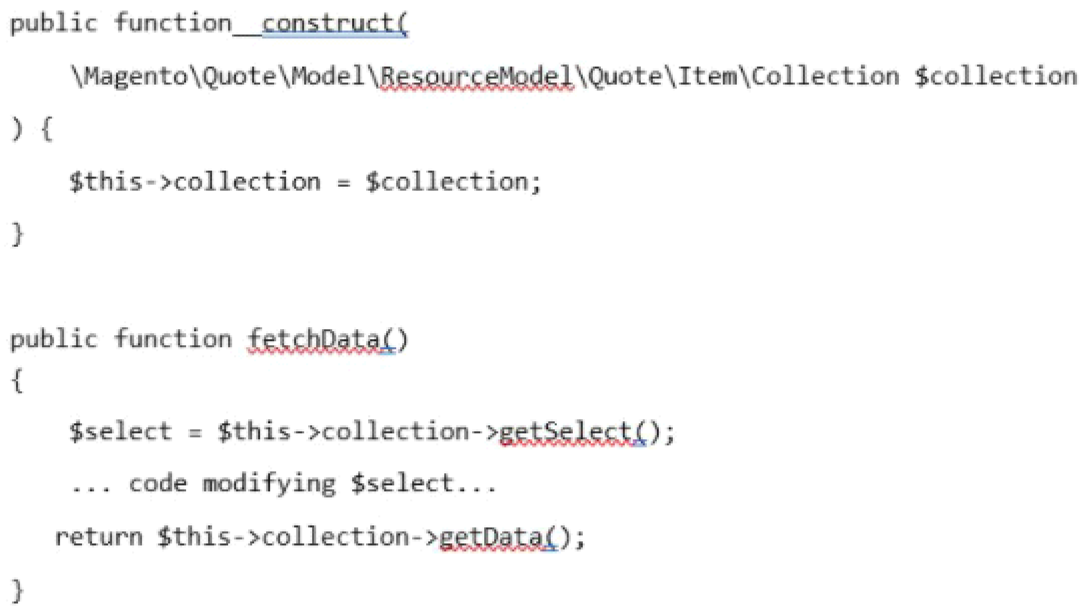
Answer:

B

Question 7

Question Type: MultipleChoice

A custom module is performing an optimized custom query for quote items. The class applies the query customizations on the select object of a quote item collection instance.



```
public function__construct(
    \Magento\Quote\Model\ResourceModel\Quote\Item\Collection $collection
) {
    $this->collection = $collection;
}

public function fetchData()
{
    $select = $this->collection->getSelect();
    ... code modifying $select...
   return $this->collection->getData();
}
```

You are tasked to resolve an issue where the query sometimes does not deliver the expected results. You have debugged the problem and found another class is also using a quote item collection and is loading the collection before the custom module. How do you resolve the issue, keeping maintainability in mind?

Options:

A- You change the argument type to \Magento\Quote\Model\ResourceModel\Quote\Item\CollectionFactory and instantiate the collection using $collectionFactory->create();

B- You remove the constructor argument and use ObjectManager::getInstance()->create(\Magento\Quote\Model\ResourceModel\Quote\Item\Collection::class) to instantiate the collection instead.

C- You inject \Magento\Framework\DB\Select instead of the collection and perform the desired query independently of the collection.

D- You inject \Magento\Quote\Api\CartItemRepositoryInterface because low level query customizations are not allowed.

Answer:

A

Question 8

Question Type: **MultipleChoice**

You are implementing a customization of the sales management within a module MyCompany_MySalesProcess. You have created several event observers to add the custom functionality. Each observer is a separate class, but they require some common functionality. How do you implement the common functionality in the event observers, keeping maintainability and testability in mind?

Options:

A- You create a trait with the common methods and use the trait in the observer classes.

B- You create an abstract class AbstractObserver with the common methods and extend the observer classes from it.

C- You create a regular class implementing the common functionality as public static methods and call those from the observers.

D- You create a regular class implementing the common functionality as public methods and use constructor injection to make them available to the observers.

Answer:

D