Demonstrate ability to use products and product types

Identify/describe standard product types (simple, configurable, bundled, etc.).

Product type model - Magento\Catalog\Model\Product\Type\AbstractType

7 product types:

- *virtual*

  Not tangible products - services, memberships, warranties, and subscriptions. No weight, can't be shipped.

- *simple*

  Single SKU, has weight and can be shipped.

- *downloadable*

  Same as virtual + downloadable links

- *configurable*

  Add 1 composite product, consisting of 1 selected variation. All options are connected. Child product inventory tracked separately.

- *grouped*

  Groups multiple products on same page. Customer can buy all of some of them, edit qty. After adding to cart, they are separate, same as if you added each of them one by one.

- *bundle*

  Add 1 composite product, consisting of many selected variations, options independent. SKU dynamic/fixed, weight dynamic/fixed, price dynamic/fixed. Each qty x total qty. Inventory is tracked separately for each child.

- *gift card (Commerce edition)*

  Like downloadable, purchase gift card code instead of link. Can be virtual/physical/both. Thus can be shipped.

product_types.xml - maximum configuration:

```
<type name="mytype" label="My Type Product" modelInstance="..." composite="false"
isQty="false" canUseQtyDecimals="true" indexPriority="10" sortOrder="10">
    <priceModel instance="...\Model\Product\Type\Mytype\Price" />
    <indexerModel instance="...\Model\ResourceModel\Product\Indexer\Price\Mytype"
/>
    <stockIndexerModel instance="...\Model\ResourceModel\Indexer\Stock\Mytype" />
    <customAttributes>
        <attribute name="is_real_product" value="true"/>
        <attribute name="refundable" value="true/false"/>
        <attribute name="taxable" value="true"/>
        <attribute name="is_product_set" value="false"/>
    </customAttributes>
    <allowedSelectionTypes>
        <type name="simple" />
        <type name="virtual" />
    </allowedSelectionTypes>
</type>
<composableTypes>
  <type name="simple"/>
  <type name="virtual"/>
  <type name="downloadable"/>
</composableTypes>
```

How would you obtain a product of a specific type?

Product type code is stored in catalog_product_entity.type_id. Thus, we can use collection:
`$productCollection->addFieldToFilter('type_id', 'simple');`

What tools (in general) does a product type model provide?

_prepareProduct - given add to cart request, returns product items that will be converted to quote items

When product is added to cart:

- \Magento\Checkout\Controller\Cart\Add::execute
- \Magento\Checkout\Model\Cart::addProduct
- \Magento\Quote\Model\Quote::addProduct
- `$cartCandidates = $product->getTypeInstance()->prepareForCartAdvanced($request, $product, $process`
- `prepareForCartAdvanced` simply calls `typeInstance._prepareProduct`
- type analyzes request and initializes one or many product items
- quote converts product items to quote_items

Standard types:

- `simple._prepareProduct` - just returns [product]. one record will be saved in quote_item

- `configurable._prepareProduct` - prepares and returns [parentProduct, childProduct]. Configures parent, configures child, then adds some useful linking info to their custom options.

    - adds parent configurable product normally
    - based on param `super_attribute` selections (color=red, size=XL), loads sub product
    - parentProduct.setCustomOptions:
        - `attributes = {color: 12, size: 35}`,
        - `product_qty_{$subProduct.id} = 1`
        - `simple_product = $subProduct`
        - recursively calls subProduct.getTypeInstance._prepareProduct. this will configure child product as well
    - subProduct.setCustomOption(`parent_product_id`, ID)
- `grouped._prepareProduct`

    - loads sub products individually from `catalog_product_link` by type id LINK_TYPE_GROUPED = 3
    - subProduct[].getTypeInstance._prepareProduct

    - some meta info is added to custom options:
        - `product_type = 'grouped'`
    - all configured products are returned as separate items, not linked to any parent
- `bundle._prepareProduct` - [parentProduct, selectedChild1, …, selectedChildN]

- `downloadable._prepareProduct` - same as simple, plus additionally generates product links

    - shows error if required links not selected
    - product.setCustomOption(`downloadable_link_ids`, ‘1,2,…’)

`_prepareProduct` is mostly where product type customization are placed. Here you analyze request (`$_POST` values), validate input, can set prices, custom option. Primary goal is to return configured products with custom options (`product.setCustomOption`). Each product will be converted to `quote_item` records, custom options will be converted to `quote_item_option` records.

processBuyRequest()

Convert buyRequest back to options to configure when you added product to cart and click Configure.

checkProductBuyState()

Check product has all required options. read product.getCustomOption, throws exception on error. Used by quote_item.checkData() - checks if product in cart is healthy.

getOrderOptions()

Prepare additional options/information for order item which will be created from this product.

- product_calculations - parent/child
- shipment_type - together/separately

getSku(), *getOptionSku()*

Glues SKU parts for custom options, bundle selections etc.

getRelationInfo() - [table, parent_field_name, child_field_name, where]

Used in:

- fulltext search to get attributes of product+children
- product flat indexer - insert children into flat

getWeight()

- configurable product used simple product’s weight
- bundle product composes sum weight of components (if weight type dynamic)

isSalable()

- by default true if product status is enabled
- data[‘is_salable’] can force not salable (data attribute set by inventory from stock index?)
- configurable - at least one enabled child product must be in stock
- bundle - all required options are salable
- downloadable - must have links

processBuyRequest()

Used for setting product preconfigured values - color, size, bundle selections - when configuring added product.

Helper\Product.prepareProductOptions:

```
$optionValues = $product->processBuyRequest($buyRequest);
  // $type->processBuyRequest
$optionValues->setQty($buyRequest->getQty());
$product->setPreconfiguredValues($optionValues);
```

- configurable - assigns super_attribute selection.
- gift card - assigns amount, messages, emails etc. - all form data from remembered buy request.