```cpp
EditItemID Edit::createNewItemID (const std::vector<EditItemID>& idsToAvoid) const
{
    // TODO: This *may* be slow under heavy load - keep an eye open for this
    // in case a smarter caching system is needed
    auto existingIDs = EditItemID::findAllIDs (state);

    existingIDs.insert (existingIDs.end(), idsToAvoid.begin(), idsToAvoid.end());
    existingIDs.insert (existingIDs.end(), usedIDs.begin(), usedIDs.end());

    trackCache.visitItems ([&] (auto i)  { existingIDs.push_back (i->itemID); });
    clipCache.visitItems ([&] (auto i)   { existingIDs.push_back (i->itemID); });

    std::sort (existingIDs.begin(), existingIDs.end());
    auto newID = EditItemID::findFirstIDNotIn (existingIDs);
    jassert (usedIDs.find (newID) == usedIDs.end());
    usedIDs.insert (newID);

    return newID;
}
```

```
EditItemID Edit::createNewItemID (const std::vector<EditItemID>& idsToAvoid) const
{
    if (nextID == 0)
    {
        auto existingIDs = EditItemID::findAllIDs (state);

        existingIDs.insert (existingIDs.end(), idsToAvoid.begin(), idsToAvoid.end());

        trackCache.visitItems ([&] (auto i)  { existingIDs.push_back (i->itemID); });
        clipCache.visitItems ([&] (auto i)   { existingIDs.push_back (i->itemID); });

        std::sort (existingIDs.begin(), existingIDs.end());
        nextID = existingIDs.empty() ? 1001 : (existingIDs.back().getRawID() + 1);

       #if JUCE_DEBUG
        usedIDs.insert (existingIDs.begin(), existingIDs.end());
       #endif
    }

    auto newID = EditItemID::fromRawID (nextID++);

   #if JUCE_DEBUG
    jassert (usedIDs.find (newID) == usedIDs.end());
    usedIDs.insert (newID);
   #endif

    return newID;
}
```

```
std::vector<EditItemID> EditItemID::findAllIDs (const juce::ValueTree& v)
{
    std::vector<EditItemID> ids;

    IDRemapping::visitAllIDDecls (v, [&] (const juce::var& oldID)
    {
        auto i = EditItemID::fromVar (oldID);

        if (i.isValid())
            ids.push_back (i);
    });

    return ids;
}
```

```
template <typename Visitor>
static void visitAllIDDecls (const juce::ValueTree& v, Visitor&& visitor)
{
    for (int i = 0; i < v.getNumProperties(); ++i)
    {
        auto propName = v.getPropertyName (i);

        if (isIDDeclaration (propName))
            visitor (v.getProperty (propName));
    }

    for (const auto& child : v)
        visitAllIDDecls (child, visitor);
}
```

52

```
EditItemID Edit::createNewItemID (const std::vector<EditItemID>& idsToAvoid) const
{
    // TODO: This *may* be slow under heavy load - keep an eye open for this
    // in case a smarter caching system is needed
    auto existingIDs = EditItemID::findAllIDs (state);

    existingIDs.insert (existingIDs.end(), idsToAvoid.begin(), idsToAvoid.end());
    existingIDs.insert (existingIDs.end(), usedIDs.begin(), usedIDs.end());

    trackCache.visitItems ([&] (auto i)  { existingIDs.push_back (i->itemID); });
    clipCache.visitItems ([&] (auto i)   { existingIDs.push_back (i->itemID); });

    std::sort (existingIDs.begin(), existingIDs.end());
    auto newID = EditItemID::findFirstIDNotIn (existingIDs);
    jassert (usedIDs.find (newID) == usedIDs.end());
    usedIDs.insert (newID);

    return newID;
}

std::vector<EditItemID> EditItemID::findAllIDs (const juce::ValueTree& v)
{
    std::vector<EditItemID> ids;

    IDRemapping::visitAllIDDecls (v, [&] (const juce::var& oldID)
    {
        auto i = EditItemID::fromVar (oldID);

        if (i.isValid())
            ids.push_back (i);
    });

    return ids;
}


    template <typename Visitor>
    static void visitAllIDDecls (const juce::ValueTree& v, Visitor&& visitor)
    {
        for (int i = 0; i < v.getNumProperties(); ++i)
        {
            auto propName = v.getPropertyName (i);

            if (isIDDeclaration (propName))
                visitor (v.getProperty (propName));
        }

        for (const auto& child : v)
            visitAllIDDecls (child, visitor);
    }
```

```
EditItemID Edit::createNewItemID (const std::vector<EditItemID>& idsToAvoid) const
{
    if (nextID == 0)
    {
        auto existingIDs = EditItemID::findAllIDs (state);

        existingIDs.insert (existingIDs.end(), idsToAvoid.begin(), idsToAvoid.end());

        trackCache.visitItems ([&] (auto i)  { existingIDs.push_back (i->itemID); });
        clipCache.visitItems ([&] (auto i)   { existingIDs.push_back (i->itemID); });

        std::sort (existingIDs.begin(), existingIDs.end());
        nextID = existingIDs.empty() ? 1001 : (existingIDs.back().getRawID() + 1);

       #if JUCE_DEBUG
        usedIDs.insert (existingIDs.begin(), existingIDs.end());
       #endif
    }

    auto newID = EditItemID::fromRawID (nextID++);

   #if JUCE_DEBUG
    jassert (usedIDs.find (newID) == usedIDs.end());
    usedIDs.insert (newID);
   #endif

    return newID;
}
```