full **step-by-step guide** for handling **merge conflicts** when **Dev1 and Dev2 are working on the same file** (`CheckoutPage.java`) and trying to merge to `master`.

Scenario Summary

- **Both Dev1 and Dev2** made changes to `CheckoutPage.java`.
- **Both are trying to merge to `master`**.
- **A merge conflict occurs**.
- Finally, Dev2's code is chosen to be merged into `master`.

Step-by-Step Guide (with commands)

Step 1: Dev1 and Dev2 work on branches

```
# Dev1
git checkout -b dev1_checkoutpage

# Dev2
git checkout -b dev2_checkoutpage
```

They both modify `CheckoutPage.java`, commit the changes, and push:

```
git add CheckoutPage.java
git commit -m "Dev1 updated CheckoutPage"
git push origin dev1_checkoutpage
```

```
git add CheckoutPage.java
git commit -m "Dev2 updated CheckoutPage with better coverage"
git push origin dev2_checkoutpage
```

Step 2: Dev1 tries to merge into master

```
git checkout master
git pull origin master
git merge dev1_checkoutpage
# If no conflicts:
git push origin master
# If conflict happens:
# Git shows: CONFLICT (content): Merge conflict in CheckoutPage.java
```

◇ Step 3: Dev2 pulls latest master and tries to merge

```
git checkout dev2_checkoutpage
git pull origin master
git merge master
# This causes a conflict in CheckoutPage.java
```

Conflict Handling Starts

Git will show:

```
Auto-merging CheckoutPage.java
CONFLICT (content): Merge conflict in CheckoutPage.java
Automatic merge failed; fix conflicts and then commit the result.
```

☑ Step 4: View and resolve the conflict manually

Edit `CheckoutPage.java` and look for conflict markers like:

```
<<<<<<< HEAD
// Dev2's changes
public void fillCheckoutForm() {
    // Updated logic by Dev2
}
=======
public void fillCheckoutForm() {
    // Dev1's older logic
}
>>>>>>> master
```

Since **Dev2's code is preferred**, remove Dev1's block and conflict markers:

```
public void fillCheckoutForm() {
    // Updated logic by Dev2
}
```

☑ Step 5: After resolving conflict

```
git add CheckoutPage.java
git commit -m "Resolved merge conflict by keeping Dev2’s changes"
git push origin dev2_checkoutpage
```

🔁 Step 6: Merge Dev2’s branch to master

```
git checkout master
git pull origin master
git merge dev2_checkoutpage
git push origin master
```

☑ Now Dev2’s changes are in master.

❌ To Abort a Merge (if needed)

If you are mid-merge and want to cancel:

```
git merge --abort
```

- ◇ It will **undo** the merge and return to the state before merge started.

🗒 Summary

Action	Command Example
Checkout new branch	`git checkout -b dev2_checkoutpage`
Commit changes	`git add . && git commit -m "Dev2 updated CheckoutPage"`
Merge to master	`git checkout master && git merge dev2_checkoutpage`
Handle conflict	Edit files manually and `git add`
Commit after resolving conflict	`git commit -m "Resolved conflict"`
Abort merge	`git merge --abort`
