Code Refactoring

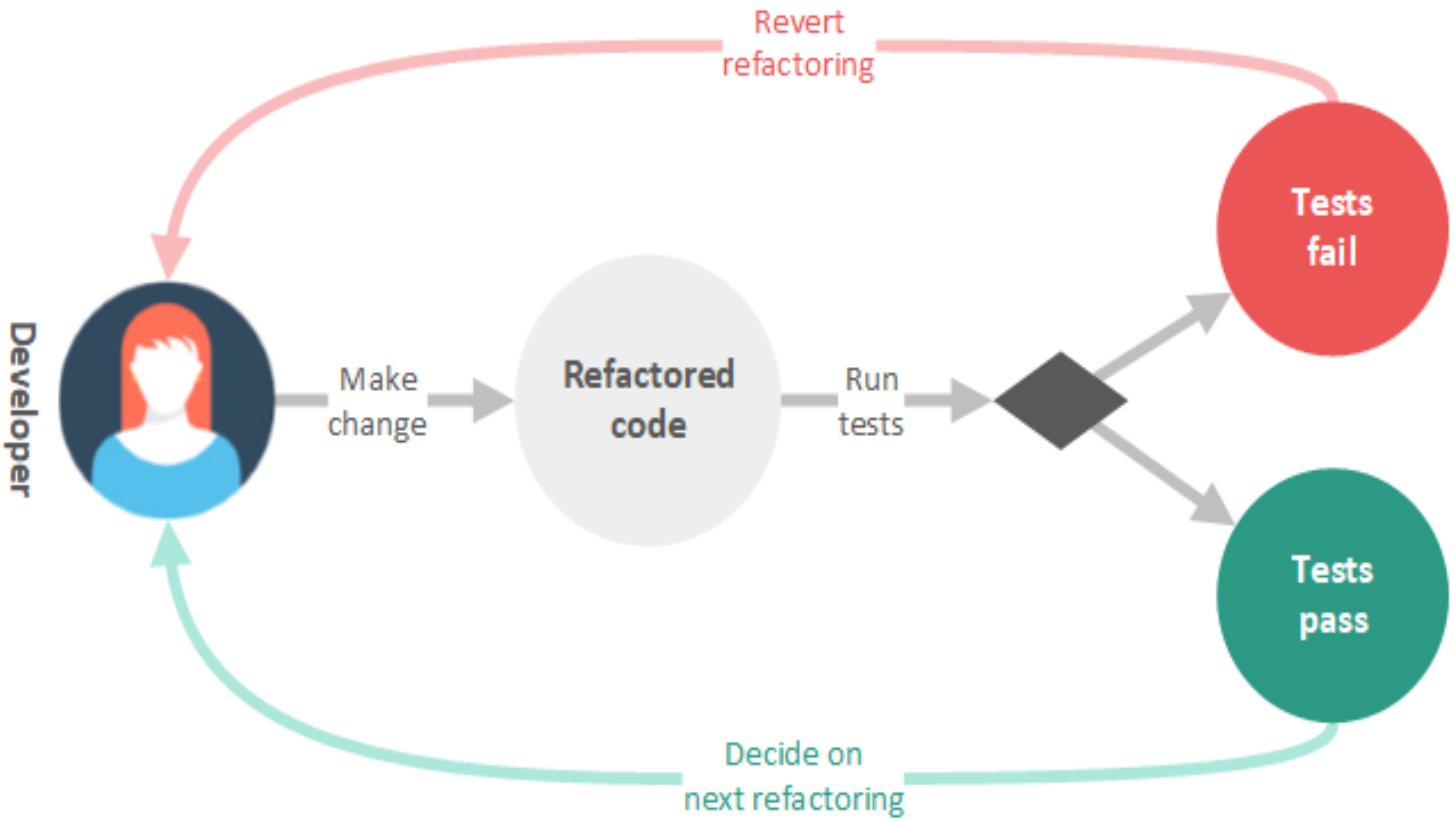
Presented by

Sagar
Java Consultant

Code Refactoring

- Refactoring is a powerful Agile technique for improving existing software.
- It helps to ensure a system is maintainable and extensible.
- The system is more maintainable
- Leads to reduced costs.
- At the same time, if the code is well structured, new requirements can be introduced more efficiently and with less problems.

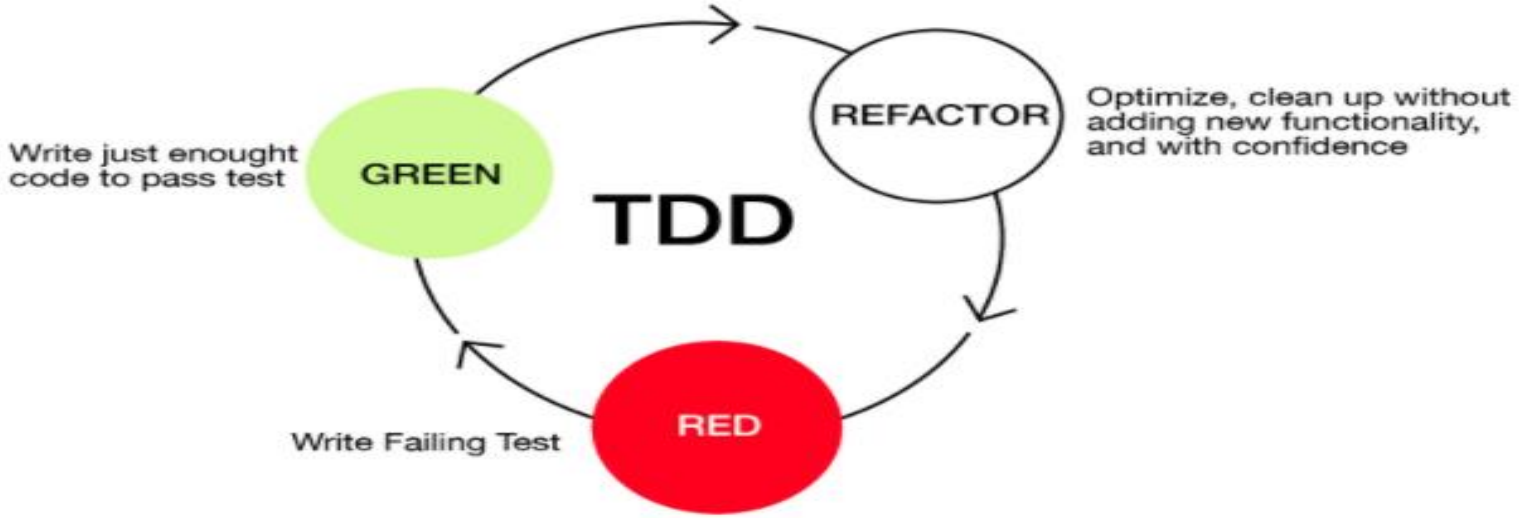
Code Refactoring …



Red-Green-Refactor

- One of the most widely used techniques for code refactoring is the red/green process used in Agile test-driven development.

1. Stop and consider what needs to be developed. [RED]
2. Get the development to pass basic testing. [GREEN]
3. Implement improvements. [REFACTOR]



Refactoring – Best Practices

- Refactor first before adding any new features
- Plan your refactoring project and timeline carefully
- Test often
- Get QA team involved
- Try refactoring automation
- As with most processes, the more it can be automated the easier and faster refactoring becomes.
- There are many shortcuts and tools to make refactoring less painful.
  - Ex: Auto indentation , main() method template, automatic setting and getters and constructor in Eclipse

Refactoring – Example

```
void printOwing() {
  printBanner();
  //print details
  System.out.println ("name: " + _name);
  System.out.println ("amount      " + getOutstanding());
  }
```

Refactored to

```
void printOwing() {
  printBanner();
  printDetails(getOutstanding());
}
void printDetails (double outstanding) {
  System.out.println ("name: " + _name);
  System.out.println ("amount      " + outstanding);
}
```


enter
return
THANK YOU
alt
control