The Software Development Life Cycle

- Gather Requirements
- System Analysis
- Design
- Implementation
- Testing
- Maintenance
Gather Requirements

Find out what the customer wants
Be sure you understand the real requirements, they may be unstated
System Analysis

Restate the requirements in software engineering and software system terms
Design

Develop a solution that satisfies the requirements developed previously.

You should check with the customer that the behavior of the system really meets the requirements of the actual customer.

A prototype is very useful at this stage.
Implementation

Now the theory is that the design is implemented in code. This is where defects in the design surface and should be addressed and not "covered up"
Testing

Verifying that the code actually meets the requirements can be hard

Verifying that the code meets the customer's intended requirements is harder and requires the assistance of the customer
Maintenance

Code is never really finished - it is either abandoned or revised to meet new requirements or fix "bugs" that are discovered "in the field"
Extreme Programming or XP

Conventional software process has some problems

- Software has a tendency to get out of control
- Generally cannot evaluate large projects until they are almost done
- Testing and maintenance plans generally forgotten in the "crunch" of getting the project done
- Meeting actual customer requirements is the exception

Many of these problems are traceable to incomplete or poorly done requirements gathering and system design

This is because the feedback loop from the design and code back to the requirements takes too long

More design and analysis up front will save money in the long run - if it is properly focused

  - must avoid analysis paralysis

Extreme Programming was designed to address these issues (and more)
Extreme Programming Principles and Practices

XP is based on several principles
- Rapid feedback
- Assume simplicity
- Incremental change
- Embrace change
- Quality work

These principles guide the XP practices
- Planning game
- Small releases
- Metaphor
- Tests
- Pair programming
- Refactoring
- Simple design
- Collective ownership
- Continuous integration
- Open workspace
- 40 hour week
Planning game
   Based on user "stories" - a lightweight form of use cases
   2 to 3 sentences that
      a customer cares about
      can be reasonably tested
      can be estimated
      can be prioritized
   Must be done with the actual customer
Tests

Unit tests
created by developer
before and during programming
all unit tests must be passed 100% of the time

Functional tests are specified by the user
run at least once a day
allow customer to know there is real progress
these tests become part of the specification
Simple design

always use "the simplest thing that could possibly work"
"you aren't going to need it" (so why write it)
If an aspect is too complicated to know what would work then a "spike" is implemented to help make the decision
Pair programming

All production code is written by a pair
Pairs are regularly switched
Many people learn many aspect of the system
Customer and Developer rights

Customer Bill of Rights - You have the right to:

- declare the business priority of every UserStory
- an overall plan, to know what can be accomplished, when, and at what cost.
- get the most possible value out of every programming week.
- see progress in a running system, proven to work by passing repeatable tests that you specify.
- change your mind, to substitute functionality, and to change priorities without paying exorbitant costs.
- be informed of schedule changes, in time to choose how to reduce scope to restore the original date.

You can even cancel at any time and be left with a useful working system reflecting investment to date.

Developer Bill of Rights - You have the right to:

- know what is needed, with clear declarations of priority.
- produce quality work at all times.
- ask for and receive help from peers, superiors, and customers.
- make, and update your own estimates.
- accept your responsibilities instead of having them assigned to you.

FortyHourWeeks (It is already part of practice: Sustainable pace)