1) Bugs, fixes:

1. Pragma solidity without ^ mark.
2. IMO mapping balance to uint248 is redundant and it makes code less readable. If I have missed some usages of converting balance to uint248 (i.e bytecode related topic or other that I have missed) then sorry for that ;)
3. Lack of event about deposing ether.
4. Lack of event about withdrawing ether.
5. Emit logs about deposit and withdraw ether.
6. Although piggyBank function uses msg.value, it is not payable.
7. Not specified visibility in functions.
8. keccak256(arg1, arg2) should wrap arguments inside keccak256(abi.encodePacked(arg1, arg2))
9. Instead of
   if (msg.sender != owner) revert();
   I think it should be:
   require(msg.sender == owner, "Only owner");
10. Instead of
    if (keccak256(owner, password) != hashedPassword) revert();
    I think it should be:
    require(keccak256(abi.encodePacked(owner, pasword) == hashedPassword, "wrong password");
11. Variables should have public access to have an easier access to their values.
12. Any contract is able to call constructor method, solution for it is to use constructor instead of function like this:
    constructor () public payable {
        owner = msg.sender;
        balance += uint248(msg.value)
    }
13. After creating such constructor function piggyBank() should be renamed to deposit()
14. If we add modifier onlyIfOwner to kill function and fallback function, password variable and parameter in contract is redundant. We could only rely on modifier

    modifier onlyIfOwner {
      require(msg.sender == owner, "Only an owner has an access");
      _;
    }
    and removing from contract
    a) param _hashedPassword from piggyBank function
    b) variable hashedPassword
    c) param password from kill function

15. Execution of kill function should be restricted to an authorized user, or user with some privileges
16. If contract has a functionality that receives ether, it must allow users to withdraw this deposited ether from the contract.

2) Bugs, fixes:

1. Missing event for shipping
2. Missing emitting Event for shipping
3. Missing payable keyword in function purchase()
4. wallet.send doesn't verify if sending ether goes successfully, we can do it by check if send goes successfully and throws if it doesn't, but I'd prefer to use transfer, as it check for us and makes code more readable. This step is required to check if funds are being handled safely.
5. No matter if we stick to send or transfer, it should be placed as a last operation in purchase method
6. In Solidity 0.4.5 interfaces were not available, therefore we should upgrade Solidity to newer version or get rid of interfaces with indication of changing version of solidity.
7. ship() function should be external.
8. purchase() function should be public.
9. Change function Store(address _wallet, address _warehouse) to constructor(address _wallet, address _warehouse)

3) Bugs, fixes:

1. Upgrade solidity to ^0.4.24
2. function Splitter(address _two) should be renamed to constructor(address _two)
3. Same function should have payable keyword in name if it uses msg.value
4. And defined visibility, in that case public
5. Fallback function should have defined visibility keyword (public)
6. Instead of
   `if (msg.value > 0) revert();`
   it should be
   `require(msg.value < 0, "No ether allowed");`
7. This contract doesn't handle modulo of dividing contract balance by 3, one of solutions is to store it in contract or refund.
8. It would be better to store addresses in address array (address[2] recipients)
9. Instead of working on `one` and `two` variable, we can work on mentioned above address array.

10. Instead of sending ether to both recipients, each of recipient balance should be stored and allow them to claim balance manually. In other case, if first recipient reject payment then contract would stop execution, therefore second recipient won’t get paid. Fallback function wrongly assume that recipient isn’t contract.
11. The target of a call instruction is vulnerable for attacker’s manipulation. Right now this contract allows an user to execute untrusted code on behalf of the contract.
12. Fallback function execution should be somehow restricted to an authorized set of users.