Contract 1 (PiggyBank):

* The balance isn’t send on kill. So the user will just lose the balance. The transfer of money should be done by owner.transfer(balance)
* There’s no check to make sure that the user killing the contract is the owner. If a user was somehow able to figure out the password that gets hashed to hashedPassword, they could kill the concert.
* Contracts already have a balance variable that is type uint256. The contract should just use this variable instead of the declared uint248 one.
* The constructor is not payable yet references msg.value. Either the constructor should be payableor that line should be gotten rid of

Contract 2 (Store):

* The purchase method needs to be payable to accept messages with value
* There is no check on the value sent to the store in the purchase method. An attacker could send 1 wei and then receive the item from the warehouse.
  + This can be done by either maintaining an internal record of how much each item costs or by communicating with a trusted warehouse contract that keeps track of price
* Maybe the Warehouse takes care of this and I can’t tell since the interface is just a black box, but I don’t see a way that delivery address gets associated with a certain customer.

Contract 3 (Splitter):

* There is no way to withdraw money from the splitter. That means that ⅓ of every transaction would be stuck there.
* The first line of the constructor isn’t needed since the function will automatically fail if it’s sent value since it isn’t payable.
* The function calculates amount off of the contract’s balance not the msg.value. An attacker could wait until the balance is high and then send a 1wei transaction and get ⅓ of the contract’s blance