Java Fundamentals – Coding challenge

This coding challenge is a mandatory exercise that must be completed. It will later need to be uploaded to BitBucket (after we have the GIT classes).

There is no deadline for this challenge to be completed, but we recommend doing it as soon as possible in case there is something we need to teach again.

## Implement the following:

* A class called Item,
  + With instance variables brand (String), name (String), price (int), printable (boolean), which is true by default
  + A constructor that receives 4 parameters, one for each instance variable
  + A constructor that receives 3 parameters, one for each instance variable except printable, so the default value remains unmodified
  + Override the toString method, printing to the console the description of the item (using the brand, name, and price)
* An interface called Printable
  + With a method called print that takes no parameters and returns no values
* An abstract class called Receipt
  + That implements the Printable interface
  + That has one instance variable, which is a collection of items (many objects of class Item)
  + ?A constructor that receives a collection of items and initializes its instance variable
  + A method called printItems, that takes no parameters and returns a String. The method must return a string that contains the description of every item contained in the receipt. If there are no items in the receipt, it must throw an exception with a message saying “Receipt has no items”. If there are no printable items in the receipt, it must throw an exception with the message “Receipt has no printable items”
* A subclass of Receipt, called MailReceipt
  + that has an instance variable called mail, which is a string.
  + That has a constructor that receives the items and the mail
  + It must implement the print method from the interface and print a message saying “Sending receipt to email: x” where x is the email that was set in the constructor, and also include the complete list of items in the message printed to the console. It must handle the exceptions thrown by the superclass method, and print “Error sending receipt on an email due to x”, where x is the specific message that the exception carries with it.
* A subclass of Receipt, called PaperReceipt
  + With a constant, called PAPER\_TYPE (by convention constants are written in uppercase and snakecase), with the value “A4”.
  + It must implement the constructor mandated by the superclass
  + It must implement the print method from the interface, with the same logic as the class MailReceipt, but the first message must say “printing receipt in A4”, and if an exception happens, it must print “Error printing receipt to paper due to x”, where x is the specific message that comes with the exception.
* Write a class with a main method, that instantiates different types of Receipts and Items, and calls the print method of the receipts, exercising different variations of items and receipts (receipts with no items, receipts with no printable items, receipts with printable and non printable items, etc)