	Download code from github/gigjacker/rlg-dotnet-fun and extract the Kata folder to do these exercises
1	Implement the factory pattern for a publisher's printing operation. The press can produce publications in the following formats: Paperbacks - A6 pages - soft cover - bound with gum Firmcover - B5 pages - firm cover - bound with gum Hardbacks - hardcover - some have dust jackets - novels, technical books and reference books are printed on A5 pages and bound with cor - spiral bound books --- cookbooks, B5 glossy paper --- diaries, A6 and A4 matt paper --- sketch pads, A3, A4 and B4 paper Magazines - glossy cover - A4 glossy pages - bound with gum Comics - matt cover - A4 recycled pages - bound with staples One printing press produces hardcover publications. A second machines handles all the others.
2	Part I (easy) Use the factory pattern to design a class hierarchy for recycling. There is a separate bin for each material. Glass, Paper, Tin, Plastic and Organic. Your design should show specific items of each material being contained in the appropriate receptacle. Part 2 (not too bad) Write a loop to get input from the console to select a specific item and report back the bin it should go in. Use polymorphism to identify the correct bin. Part 3 (not so easy) Create a list of items to be recycled and populate it with a random selection of items. Loop thru the resultant list, using polymorphism to identify the correct bin.

3	Part I (easy) Use the factory pattern to design a class hierarchy for recycling. There is a separate bin for each material. Glass, Paper, Tin, Plastic and Organic. Your design should show specific items of each material being contained in the appropriate receptacle. Part 2 (not too bad) Write a loop to get input from the console to select a specific item and report back the bin it should go in. Use polymorphism to identify the correct bin. Part 3 (not so easy) Create a list of items to be recycled and populate it with a random selection of items. Loop thru the resultant list, using polymorphism to identify the correct bin.
4	Part 1 (easy) use the factory pattern to design a class hierarchy for the booker of a music festival. the booker has a number of venues to schedule. each is suited to a particular style of music: indie, folk, jazz & prog-rock. your design should show how artists or bands are distributed among the venues. ----- Part 2 (not too bad) each venue has different opening times and the changeover between artists varies according to the style of music. set length is 45 minutes for all artists. extend your design to reflect these requirements. ----- Part 3 (not so easy) Either by getting input from the console or by generating different artists at random, fill the schedule for the festival across all venues. each artist plays one style of music only and has a finite length of time they're able to play for. Accept or reject the artists applications based on: * reject the artists that cannot play for long enough to fill the time slot. * reject the artists that the schedule for that style cannot accommodate because it's full ----- Part 4 (pushing it) let each artist have a reputation and order them in the schedule so that the act with the biggest reputation plays last on the schedule. as new applications are encountered, move the already scheduled artists to earlier in the schedule as required. if the schedule is full, allow higher reputation acts to push the lower profile acts off the schedule. show the process taking place with messages to the console.

5	Part 1 (Easy) Use the factory pattern to create an object hierarchy for a game of draughts or backgammon. The creator is the game board and the sets of pieces are the product. Draughts has 2 sets of 12 identical pieces - one white and one black. Backgammon has 2 set of 15 identical pieces - one white, one black and a pair of dice. -------- Part 2 (Not too bad) Extend your design to include a second factory pattern, allowing the set of pieces to be the creator and the individual piece to be the product. -------- Part 3 (Not so easy) Extend your design to include chess. For chess you'll need a King, Queen, 2 Bishops , 2 Knights, 2 Rooks and 8 Pawns each for black and white. -------- Part 4 (Pushing it) Extend your design to include the playing area. By default this will be an 8 x 8 grid but in the case of backgammon, this grid is 24 x 5. Your design should provide for using the same playing area Property to implement either board so that the code referring to the playing area can be reused transparently. Display the boards in a rough representation of their real life arrangement, with the pieces at their starting positions. Use your google-fu to get those starting positions if you're not familiar with the game (s) :-) -------- Part 5 (You must be joking) (Seriously, people, I really am joking with this one. I'm only putting this here so I don't forget. A bit further down the road when we're focusing on Interface Segregation, this is gonna be a good exercise :-)) * Write an interface to manage playing each game. * Establish the commonality between your interfaces and segregate them into their abstract functions.
6	use the factory pattern to set up a class structure for different types of album. albums can be single, double, triple or boxset and each disc will either have a single track list if it's on CD or two sides if it's on vinyl. you should also take into account compilation albums such as Now that's what I call Music etc.

7	use the factory pattern to set up the finer detail of the bookshop system. each book has: - title, - author(s) - isbn - genre(s) - age range - language - media - print, audio or digital the bookshop system is able cross-reference or group the books in multiple ways

8

use 1 or more factory patterns to represent the composition of a wide variety of board games.

the list below provides the contents of a few board games, compare these lists to derive your abstract view of their components and establish their commonality. you can include any board game you like - you're not restricted to the ones listed here.

Monopoly
- Board
- 3 decks of cards
- 7 types of cash
- 2 dice
- 6 mascots/counters
- houses
- hotels

cluedo
- pads
- pencils
- a die
- character pieces
- 3 decks of cards
- murder envelope
- board

let's buy hollywood
- board
- 5 decks of cards
- 7 types of cash
- 2 sets of mascots

snakes and ladders
- board
- die
- counters

backgammon
- board/box
- 2 sets of counters
- 2 dice

trivial pursuit
- board
- question cards
- pie counters
- pie slices
- dice

cranium
- board
- 4 decks of activity cards
- plasticine
- pencils
- paper
- hourglass
- die

"