

The Enigma Machine

The Full Enigma (150 Points)

Substitution ciphers that encode a message by substituting one character for another go back at least as far as Julius Caesar, who used a rotating character scheme to encode military orders. This simple type of encryption is vulnerable to statistical attacks, however, as anyone who has solved CRYPTOGRAM puzzles can attest. In World War II, the Nazi military employed an encryption scheme that addressed this weakness of simple substitution ciphers. This scheme, implemented by typewriter-sized devices known as Enigma machines, gave the Nazis a tactical advantage that greatly contributed to their early success in the war. In fact, the eventual breaking of this coding scheme by researchers at Bletchley Park, England (including Alan Turing) is hailed as one of the turning points of the war.



Enigma machines used interchangeable rotors that could be placed in different orientations to obtain different substitution patterns. More significantly, the rotors rotated after each character was encoded, changing the substitution pattern and making the code very difficult to break. The behavior of the rotating rotors can be modeled, in a simplified form, by a device consisting of labeled, concentric rings. For example, the model below has three rings labeled with the letters of the alphabet and '#' (representing a space only for clarity on the ring).

For this assignment, you are to write a fully graphical program (Java FX) that simulates the workings of a **FULL Enigma machine**, as used by the German military.

The Encryption Technique

STEP 1 - PLUGBOARD – In the first step in the process, the Nazi's used Plug Board settings to transpose/map letters. This was a simple substitution cipher, with one letter being mapped to a corresponding letter on the "day" chart. This was in the form of 10 pairs of letters (note that 20 letters get transposed, six that do not appear were unchanged). For example, using Day 1 of the sheet on our encoding sheet... a "S" would be changed to a "Z" (and vice versa, Z goes to S), a "G" would be changed to a "T" (T goes to G), etc.

To encode 'T' – go to plugboard to get 'A'

Example Plugboard DAY 3 (from sheet)

DJ **AT** CV IO ER QS LW PZ FN BH

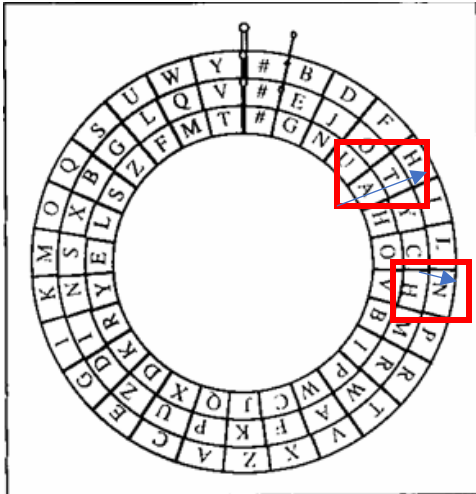


Figure 1 - SAMPLE ONLY –each wheel to map the A-Z and spaces (#=blank) 27 spaces per wheel.

STEP 2 - CIPHER WHEEL ENCRYPTION

To encrypt a character using this model, find the character on the **inner rotor** (i.e., the inside ring) and note the character aligned with it on the **outer rotor** (i.e., the outside ring), then find that character on the **middle rotor** (i.e., the middle ring) and output the one aligned with it on the **outer rotor**.

For example, in this configuration the character 'A' would be encrypted as 'N', since 'A' on the inner rotor is aligned with 'H' on the outer rotor, and 'H' on the middle rotor is aligned with 'N' on the outer rotor.

STEP 3 – REFLECTOR PANEL - The next step is to match the character with the **Reflector Panel** – a remapping of the current character, VERY SIMILAR TO THE PLUGBOARD remapping. These are represented as character pairs that map one character to another, much like a substitution cipher. For example, using the day one settings, “N” would map to “C”

Example- Reflector Panel (from sheet day 3)

KM AX PZ GO DI **CN**
BR PV LT EQ HS UW

STEP 4 – CIPHER WHEEL IN REVERSE

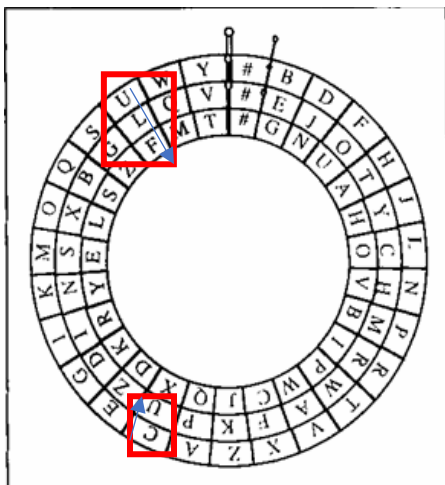


Figure 1 - SAMPLE ONLY –each wheel to map the A-Z and spaces (#=blank) 27 spaces per wheel.

After a character is mapped using the reflector panel, it is sent back through the Reflector panel **in reverse**. Find the character on the **outer rotor** and note the character aligned with it on the **middle rotor**, then find that character on the **outer rotor** and output the one aligned with it on the **inner rotor**.

Taking the C we encrypted from the previous example, - C would map to U, and U would map to F. Therefore, our output would be “F”, which goes to the final stage – the plugboard again.

STEP 5 – PLUGBOARD AGAIN

In the last step in the process, the transposed character was sent back to the plugboard to match, before getting the FINAL output. In our example, the “F” would match to “N” – so “N” would be the final output.

Example Plugboard DAY 3 (from sheet)

DJ AT CV IO ER QS LW PZ FN BH



STEP 6 – WHEEL ROTATION - After a character is encrypted, turn the **inner rotor** clockwise one-step. Whenever the inner rotor returns to its original orientation (27 ticks), the middle rotor turns once in lock-step, just like the odometer in a car. When the middle rotor turns 27 times, the outer rotor turns once in lock step.

For this assignment, you are to design a java program, and series of classes that simulate this enigma machine. Rotors consists of the 28 characters - 26 letters, plus a space, and ‘.’ (period). The rotors are interchangeable, and are given in the Enigma Machine daily encoding sheet, included on the last page.

Specifics:

- There will be five Cipher Rings. These rings will each have a set “encryption” code (but which ones to use and what starting spot will vary based on day of encoding)
-
- **ROTOR_1 = AUNGHOBIPWCJQXDKRY ELSZFM.** (note the space is 20th char, period is 28th)
ROTOR_2 = O J.ETYCHMRWAFKPUZDINSXBGLQV (note the space is 2nd char, period is 4th)
ROTOR_3 = FBDHJLNPRTVXZ.ACEGI KMQSUWY (note the space is 20th char, period is 14th)
ROTOR_4 = .HKPDEAC WTVQMYNLXSURZOJFBGI (note the space is 9th char, period is first)
ROTOR_5 = YDNGLCIQVEZRPTAOXWBMJSUH.K F (note the space is 27th char, period is 25th)
- You will implement all steps of the German encryption algorithm, as described above.
- All numbers are SPELLED out. For example, the message **Fahrenheit 451**, would be encoded as **Fahrenheit four five one.**

Use the following ACTUAL 31 Day encryption code settings sheet, on the next page, which indicates:

- **The rings to use** (inner, middle, outer), and
- **Starting spot** for each ring (1 is the first character, 28 is the last)
- **Reflector Setting**
- **PLUGBOARD** settings for the day (which pairs of letters to transpose)

For this assignment, you are given an encoded message that has the day the message was encoded, which will define the mappings for that day.

You will also encode one of your messages, provide me the day that you encode this message and include the encrypted text in your solution zip file.

German Enigma Machine setting daily sheet:

Geheime Kommandosache! Jede einzelne Tageschlüssel ist geheim. Mitlet ist im Flugzeug verboten! Nr. 00190

Luftwaffen-Maschinen-Schlüssel Nr. 649

Achtung! Schlüsselmittel dürfen nicht unversehrt in Feindeshand fallen. Bei Gefahr restlos und frühzeitig vernichten.

Tages- tag	Walzenlage			Ringstellung	Stichverordnungen										Kenngruppen				
	I	II	III		an der Umkehrwalze	1	2	3	4	5	6	7	8	9	10	1	2	3	
049 31	I	V	III	14 09 24		SZ	OT	DV	KU	FO	MY	EW	JN	IX	LQ	wny	dgy	ekb	rzg
049 30	IV	III	II	05 26 02		IS	EV	MX	RW	DT	UZ	JQ	AO	CH	NY	kti	acw	zsi	wao
049 29	III	II	I	12 24 03	KM AX FZ GO	DJ	AT	CV	IO	ER	QS	LW	PZ	FN	BH	ioc	acn	ovw	wvd
049 28	II	III	V	06 08 16	DI CN BR PV	CR	PV	AI	DK	OT	MQ	EU	BX	LP	GJ	lrb	cld	ude	rzh
049 27	III	I	IV	11 03 07	LT EQ HS UW	DY	IN	BV	GR	AM	LO	PP	HT	EX	UW	woj	fbh	vct	uis
049 26	I	IV	V	17 22 19		VZ	AL	RT	KO	CG	EI	BJ	DU	PS	HP	xle	gbo	uev	rxm
049 25	IV	III	I	08 25 12		OR	PV	AD	IT	PK	HJ	LZ	NS	EQ	CW	ouc	uhq	uew	uit
049 24	V	I	IV	05 18 14		TY	AS	OW	KV	JM	DR	HX	GL	CZ	NU	kpl	rwl	vci	tlq
049 23	IV	II	I	24 12 04	IU AS DV GL	QV	FR	AK	EO	DH	CJ	MZ	SX	GN	LT	ebn	rwm	udf	tlo
049 22	II	IV	V	01 09 21	FT OX EZ CH	PJ	ES	IM	RX	LV	AY	OU	BO	WZ	CN	jqc	acx	mwe	wve
049 21	I	V	II	13 05 19	MR KN BQ PW	RU	HL	FY	OS	QZ	DM	AW	CE	TV	NX	jpw	del	mwf	wvf
049 20	III	IV	V	24 01 10		DP	MO	QZ	AU	RY	SV	JL	GX	BE	TW	jqd	cef	nvo	ysh
049 19	V	III	I	17 25 20		OX	PR	FH	WY	DL	CM	AE	TZ	JS	GI	idf	fpz	jwg	tlg
049 18	IV	II	V	15 23 26		EJ	OY	IV	AQ	KW	FX	MT	PS	LU	BD	lsa	bw	vcj	rxn
049 17	I	IV	II	21 10 06		IR	KZ	LS	EM	OV	OY	QX	AP	JP	BU	mae	hzi	sog	ysi
049 16	V	II	III	08 16 13		HM	JO	DI	NR	BY	XZ	OS	PU	PQ	CT	tdp	dhb	fkf	uiv
049 15	II	IV	I	01 03 07	AI BT MV HU	DS	HY	MR	GW	LX	AJ	BQ	CO	IP	NT	ldw	hzj	soh	wvg
049 14	IV	I	V	15 11 05	FW EL DG KN	GM	JR	KS	IY	HZ	PL	AX	BT	CQ	NV	imz	noa	tjv	xtk
049 13	I	III	II	13 20 03	RZ OQ CP SX	LY	AG	KM	BR	IQ	JU	HV	SW	ET	CX	zgr	dgz	gjo	ryq
049 12	V	I	IV	18 10 07		MU	BP	CY	RZ	KX	AN	JT	DG	IL	PW	zdy	rkf	tjw	xtl
049 11	II	IV	III	02 26 15		KN	UY	HR	PW	FM	BO	EZ	QT	DX	JV	zea	rjy	soi	wvh
049 10	III	V	IV	23 21 01		LR	IK	MS	QU	HW	PT	OO	VX	PZ	EN	lrc	zbx	vbm	rxo
049 9	V	I	III	16 04 08		QY	BS	LN	KT	AP	IU	DW	HO	RV	JZ	edj	eyr	vby	tlh
049 8	IV	II	V	13 19 25		PI	NQ	SY	CU	BZ	AH	EL	TX	DO	KP	yiz	dha	ekc	tli
049 7	I	IV	II	09 03 22		UX	IZ	HN	BK	GQ	CP	FT	JY	MW	AR	lan	dgb	zsj	wbi
049 6	III	I	V	11 18 14	IL AP EU HO	DQ	GU	BW	NP	HK	AZ	CI	PO	JX	VY	lao	cft	zsk	wbj
049 5	V	II	IV	23 02 25	QT WZ KV GM	MV	CL	OK	OQ	BI	PU	HS	PX	NW	EY	lju	cdr	iye	waj
049 4	II	IV	I	04 21 09	BP NR DX CS	AC	BL	OZ	EK	QW	OP	SU	DH	JM	TX	lsb	zby	vcy	ujb
049 3	V	I	II	19 11 06		KR	MP	CN	BP	EH	DZ	IW	AV	GJ	LO	lap	owd	iwu	wak
049 2	IV	V	I	16 14 02		BN	HU	EG	PY	KQ	CP	OS	JW	IZ	VZ	aqd	bdy	iyf	xta
049 1	III	I	III	23 12 10		DP	BM	NZ	CK	OV	HQ	AP	UY	SW	JO	kgl	cdf	giq	wuv

Day

Which Rings to use

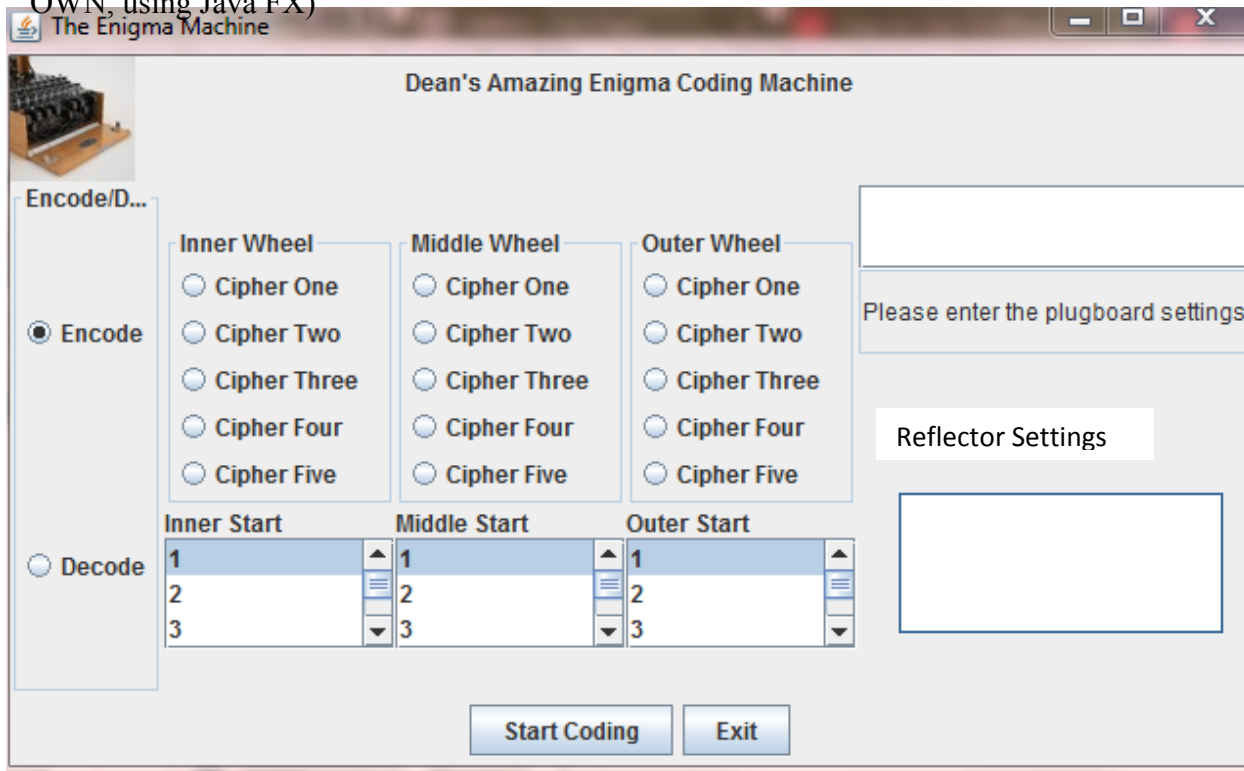
Start Position for
inner, middle, outer

Reflector Setting

Plugboard Settings

In designing your solution, you should strive to follow good object-oriented design principles. That is, define highly cohesive classes that correspond to real-world objects, and make sure those classes are loosely coupled so that the implementation details of one are independent of the others. The interface for your Enigma program is entirely up to you. It need not be fancy, but it must *at least* allow the user to select the order and settings of the rotors and subsequently encode/decode messages.

You should implement a Fully Graphical Version, as described in class... Please design your own solution, **but an example** screen shot (PLEASE BE CREATIVE AND DESIGN YOUR OWN, using Java FX)



Grading Matrix: TBD

-