Application Design Using Java

Lecture 04

Primitive Types

- Boolean
  - true or false
- Character
  - ‘a’
  - describes a *code unit* in the UTF-16 encoding

Escape sequence	Name	Unicode Value
\b	Backspace	\u0008
\t	Tab	\u0009
\n	Linefeed	\u000a
\r	Carriage return	\u000d
\"	Double quote	\u0022
\'	Single quote	\u0027
\\	Backslash	\u005c

Operators

- Precedence and Associativity: https://docs.oracle.com/javase/tutorial/java/nutsandbolts/operators.html
- Modulo: https://en.wikipedia.org/wiki/Modulo_operation

ASCII

>>> ASCII Table <<<

	0	1	2	3	4	5	6	7	8	9	A	B	C	D	E	F
0	NUL 000 0x00	SOH 001 0x01	STX 002 0x02	ETX 003 0x03	EOT 004 0x04	ENQ 005 0x05	ACK 006 0x06	BEL 007 0x07	BS 008 0x08	HT 009 0x09	LF 010 0x0A	VT 011 0x0B	FF 012 0x0C	CR 013 0x0D	SO 014 0x0E	SI 015 0x0F
1	DLE 016 0x10	DC1 017 0x11	DC2 018 0x12	DC3 019 0x13	DC4 020 0x14	NAK 021 0x15	SYN 022 0x16	ETB 023 0x17	CAN 024 0x18	EM 025 0x19	SUB 026 0x1A	ESC 027 0x1B	FS 028 0x1C	GS 029 0x1D	RS 030 0x1E	US 031 0x1F
2	SP 032 0x20	! 033 0x21	" 034 0x22	# 035 0x23	$ 036 0x24	% 037 0x25	& 038 0x26	' 039 0x27	(040 0x28	) 041 0x29	* 042 0x2A	+ 043 0x2B	, 044 0x2C	- 045 0x2D	. 046 0x2E	/ 047 0x2F
3	0 048 0x30	1 049 0x31	2 050 0x32	3 051 0x33	4 052 0x34	5 053 0x35	6 054 0x36	7 055 0x37	8 056 0x38	9 057 0x39	: 058 0x3A	; 059 0x3B	< 060 0x3C	= 061 0x3D	>>>062 0x3E	? 063 0x3F
4	@ 064 0x40	A 065 0x41	B 066 0x42	C 067 0x43	D 068 0x44	E 069 0x45	F 070 0x46	G 071 0x47	H 072 0x48	I 073 0x49	J 074 0x4A	K 075 0x4B	L 076 0x4C	M 077 0x4D	N 078 0x4E	O 079 0x4F
5	P 080 0x50	Q 081 0x51	R 082 0x52	S 083 0x53	T 084 0x54	U 085 0x55	V 086 0x56	W 087 0x57	X 088 0x58	Y 089 0x59	Z 090 0x5A	[091 0x5B	\ 092 0x5C	] 093 0x5D	^ 094 0x5E	_ 095 0x5F
6	` 096 0x60	a 097 0x61	b 098 0x62	c 099 0x63	d 100 0x64	e 101 0x65	f 102 0x66	g 103 0x67	h 104 0x68	i 105 0x69	j 106 0x6A	k 107 0x6B	l 108 0x6C	m 109 0x6D	n 110 0x6E	o 111 0x6F
7	p 112 0x70	q 113 0x71	r 114 0x72	s 115 0x73	t 116 0x74	u 117 0x75	v 118 0x76	w 119 0x77	x 120 0x78	y 121 0x79	z 122 0x7A	{ 123 0x7B	| 124 0x7C	} 125 0x7D	~ 126 0x7E	DEL 127 0x7F

44	É	160	á	176	░	192	└	208	╨	224	α	240	≡
45	æ	161	í	177	▒	193	┴	209	╤	225	ß	241	±
46	Æ	162	ó	178	▓	194	┬	210	╥	226	Γ	242	≥
47	ô	163	ú	179	│	195	├	211	╙	227	π	243	≤
48	ö	164	ñ	180	┤	196	─	212	╘	228	Σ	244	⌠
49	ò	165	Ñ	181	╡	197	┼	213	╒	229	σ	245	⌡
50	û	166	ª	182	╢	198	╞	214	╓	230	µ	246	÷
51	ù	167	º	183	╖	199	╟	215	╫	231	τ	247	≈
136	ê	152	ÿ	168	¿	184	╕	200	╚	216	╪	232	Φ	248	°
137	ë	153	Ö	169	⌐	185	╣	201	╔	217	┘	233	Θ	249	∙
138	è	154	Ü	170	¬	186	║	202	╩	218	┌	234	Ω	250	·
139	ï	155	¢	171	½	187	╗	203	╦	219	█	235	δ	251	√
140	î	156	£	172	¼	188	╝	204	╠	220	▄	236	∞	252	ⁿ
141	ì	157	¥	173	¡	189	╜	205	═	221	▌	237	φ	253	²
142	Ä	158	₧	174	«	190	╛	206	╬	222	▐	238	ε	254	■
143	Å	159	ƒ	175	»	191	┐	207	╧	223	▀	239	∩	255	

Unicode

- *Code point* is a code value that is associated with a character in an encoding scheme. Code points are written in hexadecimal and prefixed with U+, such as U+0041
- Characters in the basic multilingual plane are represented as 16-bit values, called *code units*. The supplementary characters are encoded as consecutive pairs of code units.

- *Basic multilingual plane* (code points U+0000 to U+FFFF) and 16 additional planes, with code points U+10000 to U+10FFFF, hold the *supplementary characters*
- *Surrogates area* - 2048 unused values of the basic multilingual plane (U+D800 to U+DBFF for the first code unit, U+DC00 to U+DFFF for the second code unit)
- UTF-8, UTF-16, UTF-32, etc.

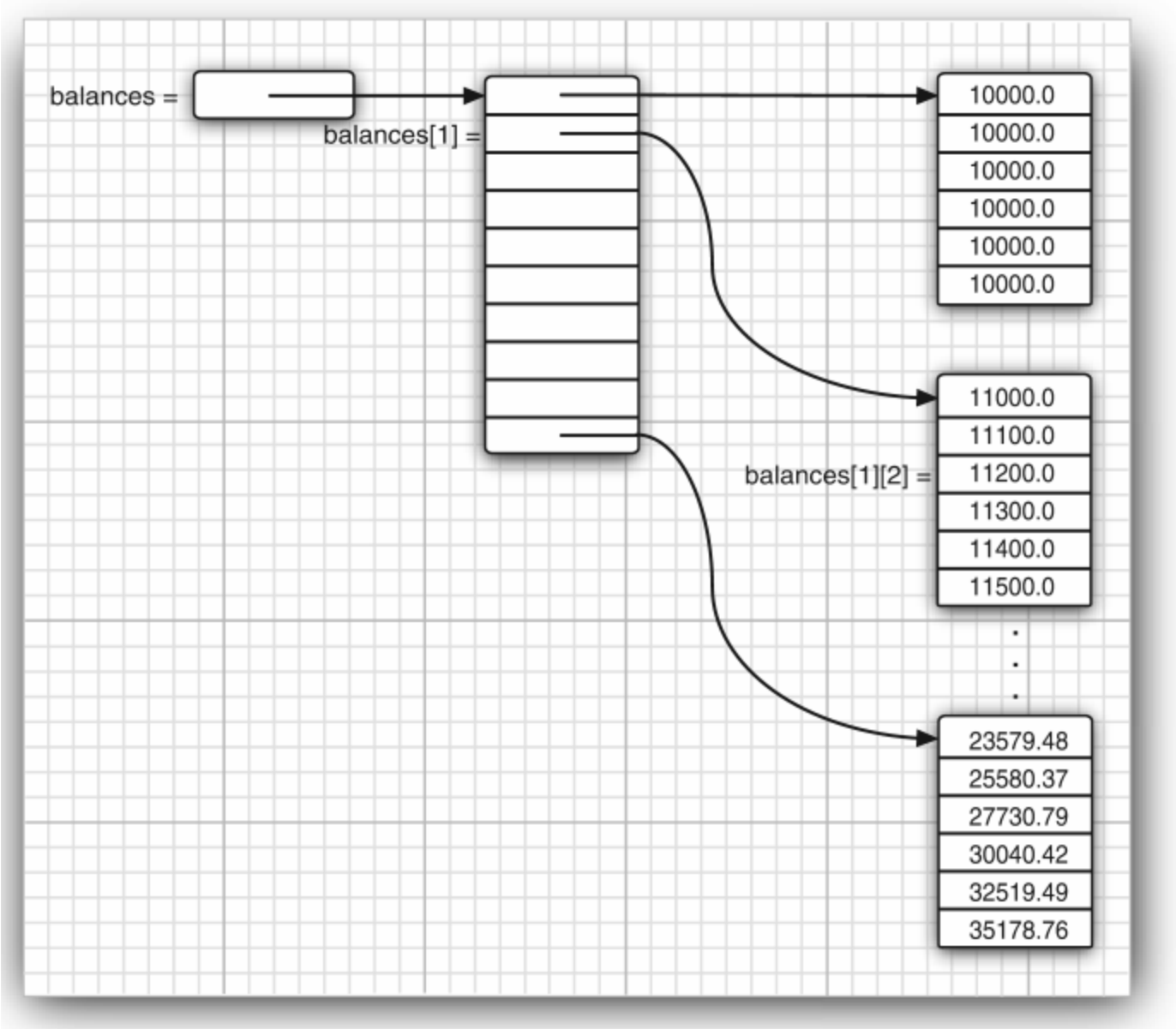
Strings

- Immutable!
- Not an array!!
- equals()!!!
- “” (str.equals("") or str.length() == 0) (but there is no empty char ‘’)
- null (str == null)
- "a", etc. Not the same as ‘a’
- substring()
- +
- length() returns the number of code units
- True length (#of code points): s.codePointCount(0, s.length())
- charAt(n) returns the code unit at position n
- To get ith code point:
  int index = s.offsetByCodePoints(0, i);
  int cp = s.codePointAt(index);
- To traverse the string:
  int cp = s1.codePointAt(i);
  if (Character.isSupplementaryCodePoint(cp)) i += 2;
  else i++;

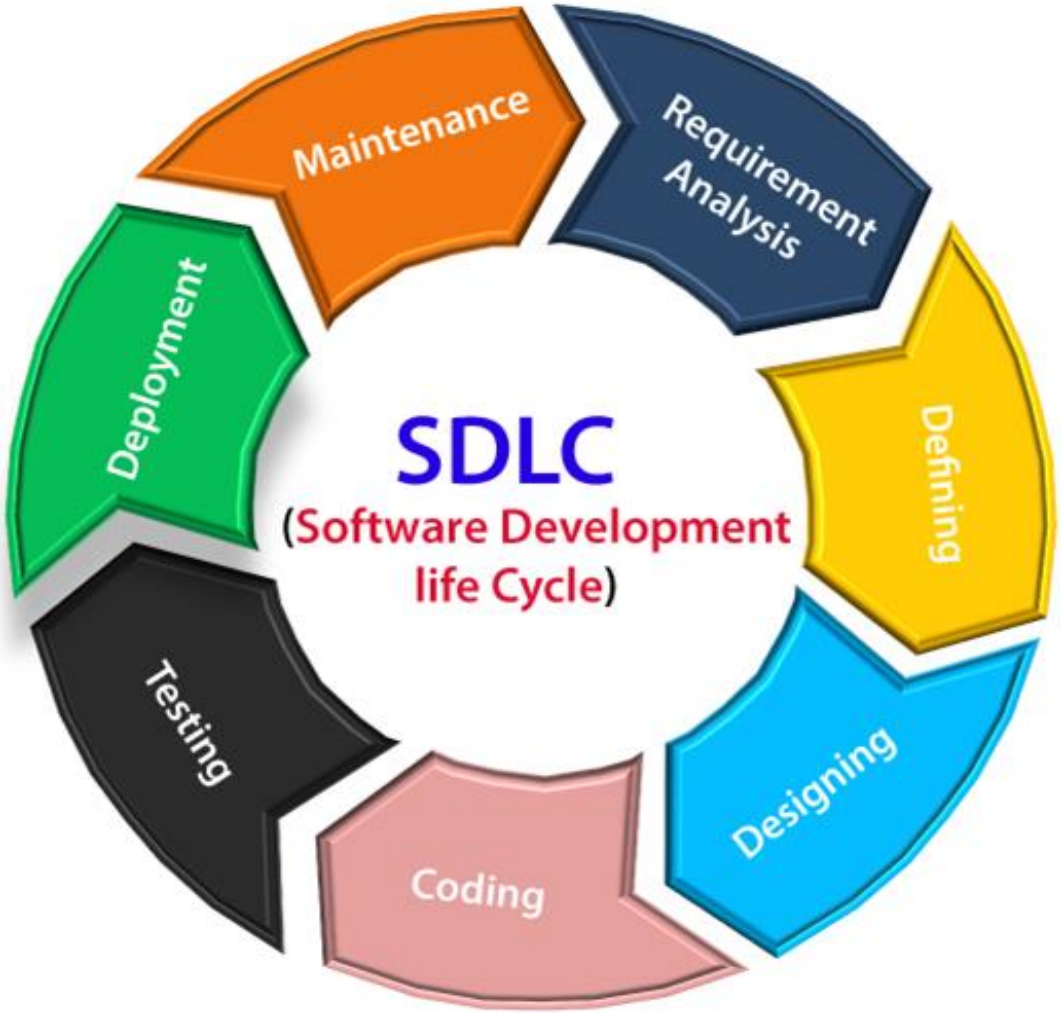
Arrays

- Regular arrays
- Ragged arrays

```
1
1  1
1  2  1
1  3  3  1
1  4  6  4  1
1  5 10 10  5 1
1  6 15 20 15 6 1
```



Software Life Cycle



Testing

CSCI-4460 Large Scale Programming and Testing

CSCI-4440 Software Design and Documentation

- Execution
  - Manual
  - Automatic
- Knowledge about implementation/internal structure
  - White box
  - Black box
- Types
  - Unit
  - Integration
  - Functional
  - Regression
  - End-to-end
  - Acceptance
  - Performance
  - …

Software Testing Life Cycle

Test Documents

- Test plan
- Test strategy
- Test scenarios
- Test case
- Requirement Traceability Matrix (RTM)
- Test data
- Bug report
- Test execution report
- …

Test Plan

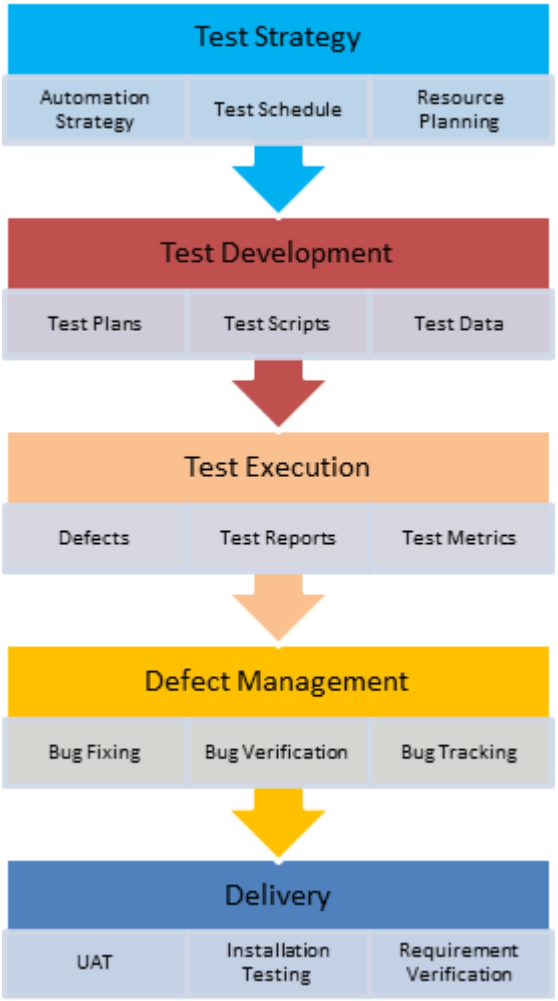
- Describes software testing areas and activities
- Outlines
  - Test strategy
  - Objectives
  - Test schedule
  - Required resources (human resources, software, and hardware)
  - Test estimation
  - Test deliverables
- Types
  - Master Test Plan
  - Phase Test Plan
  - Testing Type Specific Test Plans
- An example template in TestPlan.docx

Writing a Test Plan

- Analyze product structure and architecture
- Design the test strategy
- Define all the test objectives
- Define the testing area
- Define all the useable resources
- Schedule all activities in an appropriate manner
- Determine all the Test Deliverables

Test Strategy

- A plan for defining an approach to the STLC
- Guides QA teams to define test coverage and testing scope
- Helps testers get a clear picture of the project at any instance
- An example template in TestStrategy.docx



Test Plan vs. Test Strategy

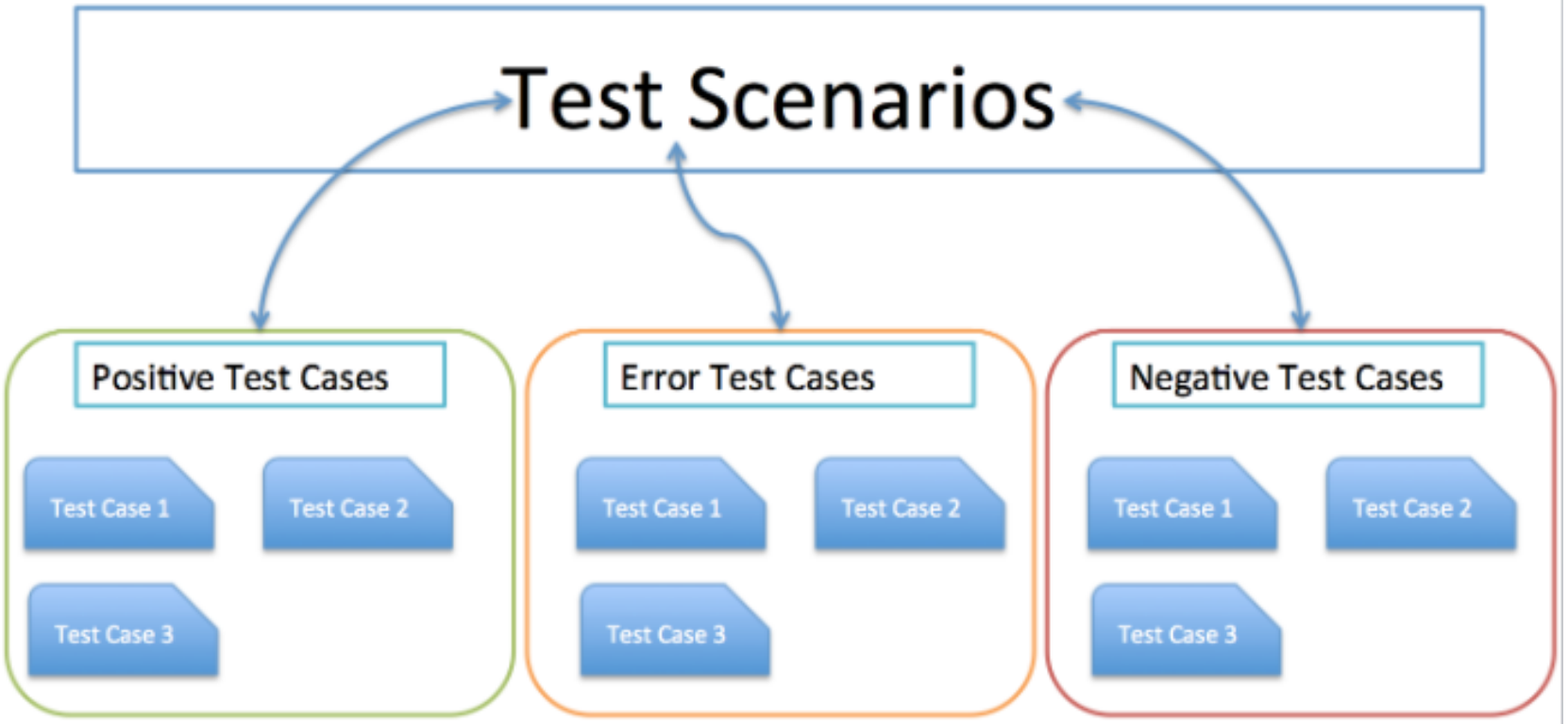
Test Plan

•In the Test Plan, test focus and project scope are defined. It deals with test coverage, scheduling, features to be tested, features not to be tested, estimation and resource management.

Test Strategy

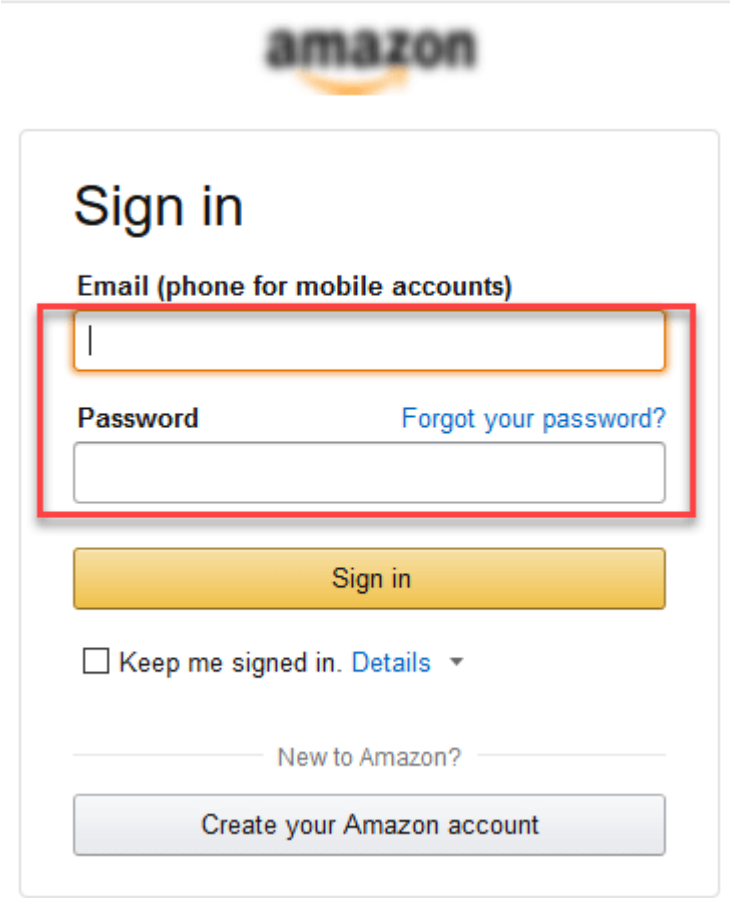
•Test strategy is a guideline to be followed to achieve the test objective and execution of test types mentioned in the testing plan. It deals with test objective, test environment, test approach, automation tools and strategy, contingency plan, and risk analysis.

Test Scenarios

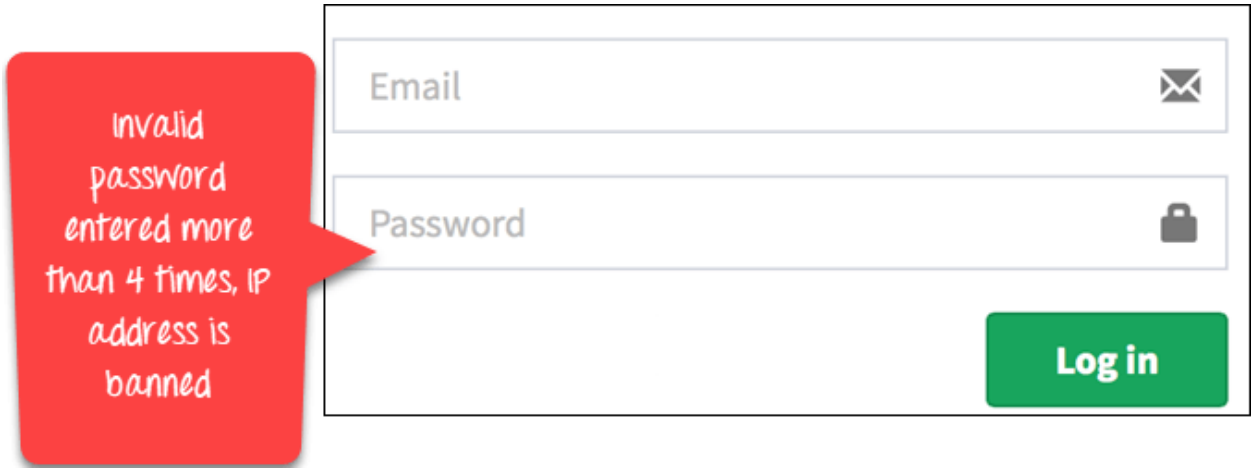


Test Scenario Example

- Test cases
  - Check system behavior when valid email id and password is entered.
  - Check system behavior when *invalid* email id and *valid* password is entered.
  - Check system behavior when *valid* email id and *invalid* password is entered.
  - Check system behavior when *invalid* email id and *invalid* password is entered.
  - Check system behavior when email id and password are left blank and Sign in entered.
  - Check Forgot your password is working as expected
  - Check system behavior when valid/invalid phone number and password is entered.
  - Check system behavior when "Keep me signed" is checked



Use Case



- A brief description of a particular use of the software application by an actor or user
- Made on the basis of user actions and the response of the software application to those user actions
- Used in developing test cases

Main Success Scenario	Step	Description
A:Actor S:System	1	A: Enter Agent Name & Password
	2	S: Validate Password
	3	S: Allow Account Access
Extensions	2a	<u>Password not valid</u> S: Display Message and ask for re-try 4 times
	2b	<u>Password not valid 4 times</u> S: Close Application

Positive and Negative Testing

- Positive - providing the valid data sets as an input
- Negative - providing invalid or improper data sets as input
- Testing techniques
  - Boundary Value Analysis
  - Equivalence Partitioning

Enter Only Numbers

99999

Positive Testing

Enter Only Numbers

abcdef

Negative Testing

Test Case

- A written document of conditions or a set of variables through which a tester examines whether software is fulfilling all requirements or not.
- Test scenarios are rather vague and cover a wide range of possibilities. Test cases are very specific.
- Correlates with a use case
- Fields
  - Test Case ID
  - Title
  - Priority
  - Description
  - Steps
  - Expected Result
  - Status
- Either “passes” or “fails”
- An example template in TestCaseTemplate.xls

//TODO before next lecture:

- Homework 1 has been posted. Stat working on it, if you haven't already. It is due on Friday next week, 2/12, at 11:59 pm EST. Must be submitted on Submitty.
- Challenge question:
  - You can use a trick to convert a Boolean to an integer if you really want to, with a single expression. Can you guess how?
- Java puzzler (posted on Submitty Forum)