IIC 2143 Ingeniería de Software
Examen - Semestre 1 /2019
Secciones 01 y 02

Responda cada pregunta en hoja separada
Entregue una hoja con su nombre para cada pregunta aunque sea en blanco
Para la pregunta 4, responda en la misma hoja de enunciado
Tiempo: 2:30
Recuerden que están bajo el código de honor

Pregunta 1:

- En ingeniería de software, decimos que un proyecto es exitoso si logra finalizarse cumpliendo 3 tipos de restricciones: alcance, costo y calendario. En una determinada empresa, se sabe que:
  - 10% de los proyectos no cumplen las restricciones de alcance
  - 15% de los proyectos no cumplen las restricciones de costo
  - 20% de los proyectos no cumplen las restricciones de calendario

  Además, se sabe que, de los proyectos que no cumplen las restricciones de calendario, el 40% no cumple las restricciones de costo y el 30% no cumple las de alcance. De los proyectos que no cumplen ni las restricciones de calendario, ni las de costo, la mitad tampoco cumple las de alcance. Por último, sabemos que un 5% de los proyectos no cumple simultáneamente las restricciones de costo y alcance.

  a) Dado lo anterior, ¿qué probabilidad tiene un proyecto en esta empresa de tener éxito?

  (3 puntos)

- Al analizar los resultados de una sesión de Sprint Planning, un administrador se da cuenta que las horas estimadas para completar los relatos de usuario seleccionados siguen una distribución normal con media $\mu = 30$ y varianza $\sigma^2 = 36$.

  b) ¿Cuál es la probabilidad de que un relato de usuario tenga una estimación superior a las 40 horas?

  (1.5 puntos)

  c) ¿Qué valor constituye el percentil 20 de la distribución anterior?

  (1.5 puntos)

z	.00	.01	.02	.03	.04	.05	.06	.07	.08	.09
–3.4	.0003	.0003	.0003	.0003	.0003	.0003	.0003	.0003	.0003	.0002
–3.3	.0005	.0005	.0005	.0004	.0004	.0004	.0004	.0004	.0004	.0003
–3.2	.0007	.0007	.0006	.0006	.0006	.0006	.0006	.0005	.0005	.0005
–3.1	.0010	.0009	.0009	.0009	.0008	.0008	.0008	.0008	.0007	.0007
–3.0	.0013	.0013	.0013	.0012	.0012	.0011	.0011	.0011	.0010	.0010
–2.9	.0019	.0018	.0018	.0017	.0016	.0016	.0015	.0015	.0014	.0014
–2.8	.0026	.0025	.0024	.0023	.0023	.0022	.0021	.0021	.0020	.0019
–2.7	.0035	.0034	.0033	.0032	.0031	.0030	.0029	.0028	.0027	.0026
–2.6	.0047	.0045	.0044	.0043	.0041	.0040	.0039	.0038	.0037	.0036
–2.5	.0062	.0060	.0059	.0057	.0055	.0054	.0052	.0051	.0049	.0048
–2.4	.0082	.0080	.0078	.0075	.0073	.0071	.0069	.0068	.0066	.0064
–2.3	.0107	.0104	.0102	.0099	.0096	.0094	.0091	.0089	.0087	.0084
–2.2	.0139	.0136	.0132	.0129	.0125	.0122	.0119	.0116	.0113	.0110
–2.1	.0179	.0174	.0170	.0166	.0162	.0158	.0154	.0150	.0146	.0143
–2.0	.0228	.0222	.0217	.0212	.0207	.0202	.0197	.0192	.0188	.0183
–1.9	.0287	.0281	.0274	.0268	.0262	.0256	.0250	.0244	.0239	.0233
–1.8	.0359	.0351	.0344	.0336	.0329	.0322	.0314	.0307	.0301	.0294
–1.7	.0446	.0436	.0427	.0418	.0409	.0401	.0392	.0384	.0375	.0367
–1.6	.0548	.0537	.0526	.0516	.0505	.0495	.0485	.0475	.0465	.0455
–1.5	.0668	.0655	.0643	.0630	.0618	.0606	.0594	.0582	.0571	.0559
–1.4	.0808	.0793	.0778	.0764	.0749	.0735	.0721	.0708	.0694	.0681
–1.3	.0968	.0951	.0934	.0918	.0901	.0885	.0869	.0853	.0838	.0823
–1.2	.1151	.1131	.1112	.1093	.1075	.1056	.1038	.1020	.1003	.0985
–1.1	.1357	.1335	.1314	.1292	.1271	.1251	.1230	.1210	.1190	.1170
–1.0	.1587	.1562	.1539	.1515	.1492	.1469	.1446	.1423	.1401	.1379
–0.9	.1841	.1814	.1788	.1762	.1736	.1711	.1685	.1660	.1635	.1611
–0.8	.2119	.2090	.2061	.2033	.2005	.1977	.1949	.1922	.1894	.1867
–0.7	.2420	.2389	.2358	.2327	.2296	.2266	.2236	.2206	.2177	.2148
–0.6	.2743	.2709	.2676	.2643	.2611	.2578	.2546	.2514	.2483	.2451
–0.5	.3085	.3050	.3015	.2981	.2946	.2912	.2877	.2843	.2810	.2776
–0.4	.3446	.3409	.3372	.3336	.3300	.3264	.3228	.3192	.3156	.3121
–0.3	.3821	.3783	.3745	.3707	.3669	.3632	.3594	.3557	.3520	.3483
–0.2	.4207	.4168	.4129	.4090	.4052	.4013	.3974	.3936	.3897	.3859
–0.1	.4602	.4562	.4522	.4483	.4443	.4404	.4364	.4325	.4286	.4247
–0.0	.5000	.4960	.4920	.4880	.4840	.4801	.4761	.4721	.4681	.4641

z	.00	.01	.02	.03	.04	.05	.06	.07	.08	.09
0.0	.5000	.5040	.5080	.5120	.5160	.5199	.5239	.5279	.5319	.5359
0.1	.5398	.5438	.5478	.5517	.5557	.5596	.5636	.5675	.5714	.5753
0.2	.5793	.5832	.5871	.5910	.5948	.5987	.6026	.6064	.6103	.6141
0.3	.6179	.6217	.6255	.6293	.6331	.6368	.6406	.6443	.6480	.6517
0.4	.6554	.6591	.6628	.6664	.6700	.6736	.6772	.6808	.6844	.6879
0.5	.6915	.6950	.6985	.7019	.7054	.7088	.7123	.7157	.7190	.7224
0.6	.7257	.7291	.7324	.7357	.7389	.7422	.7454	.7486	.7517	.7549
0.7	.7580	.7611	.7642	.7673	.7704	.7734	.7764	.7794	.7823	.7852
0.8	.7881	.7910	.7939	.7967	.7995	.8023	.8051	.8078	.8106	.8133
0.9	.8159	.8186	.8212	.8238	.8264	.8289	.8315	.8340	.8365	.8389
1.0	.8413	.8438	.8461	.8485	.8508	.8531	.8554	.8577	.8599	.8621
1.1	.8643	.8665	.8686	.8708	.8729	.8749	.8770	.8790	.8810	.8830
1.2	.8849	.8869	.8888	.8907	.8925	.8944	.8962	.8980	.8997	.9015
1.3	.9032	.9049	.9066	.9082	.9099	.9115	.9131	.9147	.9162	.9177
1.4	.9192	.9207	.9222	.9236	.9251	.9265	.9279	.9292	.9306	.9319
1.5	.9332	.9345	.9357	.9370	.9382	.9394	.9406	.9418	.9429	.9441
1.6	.9452	.9463	.9474	.9484	.9495	.9505	.9515	.9525	.9535	.9545
1.7	.9554	.9564	.9573	.9582	.9591	.9599	.9608	.9616	.9625	.9633
1.8	.9641	.9649	.9656	.9664	.9671	.9678	.9686	.9693	.9699	.9706
1.9	.9713	.9719	.9726	.9732	.9738	.9744	.9750	.9756	.9761	.9767
2.0	.9772	.9778	.9783	.9788	.9793	.9798	.9803	.9808	.9812	.9817
2.1	.9821	.9826	.9830	.9834	.9838	.9842	.9846	.9850	.9854	.9857
2.2	.9861	.9864	.9868	.9871	.9875	.9878	.9881	.9884	.9887	.9890
2.3	.9893	.9896	.9898	.9901	.9904	.9906	.9909	.9911	.9913	.9916
2.4	.9918	.9920	.9922	.9925	.9927	.9929	.9931	.9932	.9934	.9936
2.5	.9938	.9940	.9941	.9943	.9945	.9946	.9948	.9949	.9951	.9952
2.6	.9953	.9955	.9956	.9957	.9959	.9960	.9961	.9962	.9963	.9964
2.7	.9965	.9966	.9967	.9968	.9969	.9970	.9971	.9972	.9973	.9974
2.8	.9974	.9975	.9976	.9977	.9977	.9978	.9979	.9979	.9980	.9981
2.9	.9981	.9982	.9982	.9983	.9984	.9984	.9985	.9985	.9986	.9986
3.0	.9987	.9987	.9987	.9988	.9988	.9989	.9989	.9989	.9990	.9990
3.1	.9990	.9991	.9991	.9991	.9992	.9992	.9992	.9992	.9993	.9993
3.2	.9993	.9993	.9994	.9994	.9994	.9994	.9994	.9995	.9995	.9995
3.3	.9995	.9995	.9995	.9996	.9996	.9996	.9996	.9996	.9996	.9997
3.4	.9997	.9997	.9997	.9997	.9997	.9997	.9997	.9997	.9997	.9998

Pregunta 2:

La compañía para la cual usted trabaja acaba de adquirir un software de una empresa norteamericana capaz de estimar el deterioro del motor de un vehículo según su kilometraje y la capacidad de combustible del estanque. Lamentablemente, el software fue construido con unidades imperiales en mente, por lo que no es apto para ser usado con el sistema métrico internacional de forma directa. El software ofrece una clase `EngineAnalyzer` la cual define el método `analyze_vehicle(vehicle)`, donde `vehicle` es una instancia de una clase que debiese implementar los métodos:

- `mileage`: retorna la cantidad de millas recorridas por un vehículo
- `fuel_tank_capacity`: retorna la capacidad del estanque de combustible en galones

Actualmente, usted se encuentra trabajando con un software donde ya existe una clase `Vehiculo` que implementa los métodos:

- `kilometraje`: retorna la cantidad de kilómetros recorridos por un vehículo
- `capacidad_del_estanque`: retorna la capacidad del estanque de combustible en litros

Se le pide integrar este software con la librería norteamericana sin modificar el código ya existente en la clase `Vehiculo`.

Se le recuerda que:

- 1 milla = 1.61 kilómetros
- 1 galón = 3.79 litros

a) Escriba un breve script en Ruby que pueda aplicar exitosamente el método `analyze_vehicle(vehicle)` para realizar un análisis de una instancia de la clase `Vehiculo` usando el patrón Adapter. Asuma que las clases `EngineAnalyzer` y `Vehiculo` vienen dadas, por lo que puede instanciarlas, pero no puede implementarlas.

(2 puntos)

b) Dibuje el diagrama de clases UML de su implementación.

(2 puntos)

c) Modele el flujo de su script utilizando un diagrama UML de secuencia. El primer *lifeline* de su diagrama debiese corresponder a su script.

(2 puntos)

Pregunta 3:

Una tienda online ofrece descuentos dependiendo del monto de la compra realizada por el cliente y de si la compra se hace en día de semana (Lunes a Viernes) o fin de semana (Sábado, Domingo). Si la compra es entre US$1 y US$50 no hay descuento. Si es de mas de US$50 y hasta US$200 hay un 5% de descuento. Entre US$201 y US$500 hay un 10% de descuento y de más de US$500 un 20% de descuento. Adicionalmente, en compras de Lunes a Viernes hay un 10% adicional de descuento sobre lo anterior, lo que no aplica en compras de fin de semana. Múltiples descuentos sucesivos se aplican de forma multiplicativa (ver ejemplo en inciso a)).

a) Escriba una función Ruby `calcula` que tome como input el monto de la compra y el día de semana de la compra (1 a 7, donde 1 corresponde a Lunes y 7 corresponde a Domingo), y devuelva el nuevo monto con el descuento aplicado. Por ejemplo `calcula(10, 7)` devuelve 10 pero `calcula(1000, 1)` devuelve 720 (20% y 10%).

(2 puntos)

b) Diseñe pruebas de cobertura de caminos utilizando técnicas de caja blanca (white box), de modo de tener un 100% de cobertura.

(2 puntos)

c) Diseñe los casos de prueba necesarios utilizando técnicas de caja negra (black box), según la metodología de clases de equivalencia vista en clases.

(2 puntos)

Nombre: _____

Pregunta 4:

Complete el texto con palabras seleccionadas de la lista de abajo (puede poner solo el número). Cada número se puede usar solo una vez. Si una misma opción figura dos veces en el listado, puede ser usada dos veces.

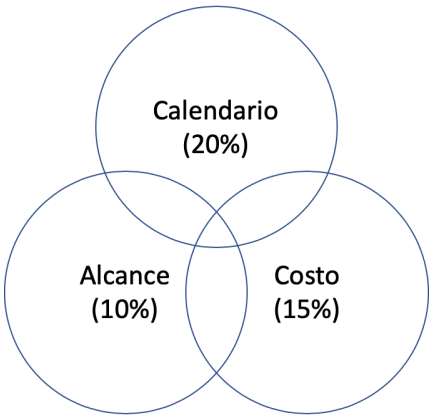
La planificación de un proyecto de software solía hacerse en forma detallada desde un principio. El resultado de ello era una _____que describía el momento de inicio y de término de cada una de las actividades. Las _____como Scrum introducen la idea de una _____donde se detallan las tareas que se realizarán en un período corto y una _____en que se planifica el número de iteraciones que existirán hasta el _____
A medida que transcurren _____ , se vuelve a revisar el plan del release y puede que requiera cambios. Lo mas probable es que el trabajo a realizar se haya _____, por lo que ya no sería posible entregar en la fecha inicialmente planeada y haya que ajustar. El ajuste puede ser _____ en que se reduce el número de features comprometidas dejando algunas menos prioritarias para _____o el ajuste puede ser _____, en que debe renegociarse la fecha de entrega para acomodar una o dos iteraciones adicionales. El primero su ser aceptado por el cliente (usuario) siempre que lo que quede fuera no sea crucial. Por ello es muy importante que quien priorice las funcionalidades que se implementan primero sea _____ y no _____. Al no ajustar ni por alcance ni por tiempo lo que suele ocurrir es un _____, al intentar de cualquier forma hacer todo lo comprometido en la fecha original. Esta situación es _____, puesto que puede producirse una _____ imposible de pagar o puede terminar entregándose un producto que no responderá a las expectativas o que presenta numerosas fallas.
En el caso de planificación de un sprint no hay posibilidad de ajustar _____ por lo que el ajuste se dará necesariamente _____. En este caso las funcionalidades planeadas no implementadas no necesariamente pasan a la siguiente iteración. Para planificar es necesario hacer _____ medido en _____y de _____, medido en _____ de desarrollo. El input para este proceso de estimación es _____ medido en _____, _____o _____. La ventaja de la primera métrica es que es fácil de registrar cuando el producto ya ha sido construido. La ventaja de los dos últimos es que no dependen del lenguaje o del estilo de programación utilizado. En el caso de los puntos de relato una técnica bastante utilizada es el llamado _____, en que cada miembro de un grupo pequeño hace _____ sobre el puntaje de cada relato hasta llegar a un consenso. También es común que los puntajes que se asignen a los relatos desde muy simple a muy complejo no aumenten en forma lineal, sino en una _____ o una _____.
Durante la época de desarrollo en cascada se elaboraron sofisticados modelos que relacionaban el tamaño con el esfuerzo o la duración (Cocomo) pero en realidad mas importante que la sofisticación del modelo es contar con _____ de proyectos anteriores similares ojalá del mismo equipo de desarrollo. Es una buena práctica hacer estimaciones de duración del proyecto que consideren tanto _____como una _____, para exponer de manera explícita el _____ de los desarrolladores. En efecto, el tiempo real suele estar mucho mas cerca del _____ que del _____.

1. ajuste por alcance
2. ajuste por calidad
3. ajuste por comprensión
4. atraso en el proyecto
5. canasta de relatos
6. carta gantt
7. casos de uso
8. datos reales propios
9. datos validados de estudios internacionales
10. deuda técnica
11. el equipo de desarrollo
12. el jefe de proyecto
13. el product owner
14. el scrum master
15. el siguiente release
16. el siguiente sprint
17. el tamaño del software a construir
18. el tiempo necesario para desarrollar el producto
19. escala cuadrática
20. escala exponencial
21. estimaciones de duración
22. estimaciones de esfuerzo
23. estimaciones de tamaño
24. exceso de optimismo
25. exceso de pesimismo
26. extremadamente riesgosa
27. fin de la iteración
28. hoja de ruta
29. hombres mes (o meses hombre)
30. las iteraciones
31. líneas de código
32. mas corto
33. mas largo
34. mas probable
35. metodologías ágiles
36. muy positiva
37. número de funciones
38. pesimista (worst case)
39. planificación a nivel de portafolio
40. planificación a nivel de producto
41. planificación a nivel de release
42. planificación a nivel de sprint
43. por alcance
44. por alcance
45. por calidad
46. por calidad
47. por tiempo
48. por tiempo
49. puntos de función
50. puntos de relato
51. release del producto
52. scrum poker
53. secuencia de fibbonacci
54. semanas (o meses)
55. serie de taylor
56. sobreestimado
57. subestimado
58. tablero kanban
59. tablero kanban
60. tiempo
61. una apuesta o predicción
62. una mirada optimista (best case

Pauta Pregunta 1:

Inciso a):

La clave para responder esta pregunta surge del análisis gráfico de los proyectos no exitosos:



Necesitamos calcular el área cubierta por los 3 círculos. Esto se puede plantear como:

$$P(Ca) + P(Co) + P(A) - P(Ca \cap Co) - P(Ca \cap A) - P(Co \cap A) + P(Ca \cap Co \cap A)$$

Tenemos que calcular estas 7 probabilidades:

- $P(Ca) = 0.2$, por enunciado
- $P(Co) = 0.15$, por enunciado
- $P(A) = 0.1$, por enunciado
- $P(Co \cap A) = 0.05$, por enunciado
- $P(Ca \cap A) = P(Ca)P(A|Ca) = 0.2 * 0.3 = 0.06$
- $P(Ca \cap Co) = P(Ca)P(Co|Ca) = 0.2 * 0.4 = 0.08$
- $P(Ca \cap Co \cap A) = P(Ca \cap Co)P(A|Ca \cap Co) = 0.08 * 0.5 = 0.04$

Reemplazando en la fórmula inicial, la probabilidad de fracaso de un proyecto es:
0.2 + 0.15 + 0.1 – 0.05 – 0.06 – 0.08 + 0.04 = 0.3

Se nos pide sin embargo la probabilidad de éxito, esto es simplemente 1 – 0.3 = 0.7

Nota de Pauta: Asignar 1 punto por plantear correctamente la fórmula que entrega la solución. Si el alumno planteó bien los 6 primeros términos de la fórmula, pero erró en la aplicación del último término (lo omitió, lo restó en vez de sumarlo…), asignar 0.5 puntos. Asignar 0.5 puntos por cada una de las 3 probabilidades no entregadas por el enunciado que había que calcular. Asignar 0.5 puntos por llegar a la respuesta correcta al final, 0 puntos en este último sub-ítem para cualquier otro resultado.

Inciso b):

Sea X la variable aleatoria que representa las estimaciones en horas de los relatos de usuario sabemos que X sigue una distribución normal tal que: $X \sim N(30,6)$. Entonces:

$$\begin{aligned} P(X > 40) &= 1 - P(X < 40) \\ &= 1 - F\left(\frac{40-30}{6}\right) \\ &= 1 - F\left(\frac{10}{6}\right) \\ &= 1 - 0.952 \\ &= 0.048 \end{aligned}$$

Inciso c):

El percentil 20 corresponde a los valores para los que F(x) = 0.2. Esto lo podemos resolver como:

$$\begin{aligned} F(z) = 0.2 &\equiv F\left(\frac{x-30}{6}\right) = 0.2 \\ &\equiv F\left(\frac{x-30}{6}\right) = F\left(F^{-1}(0.2)\right) \\ &\equiv F\left(\frac{x-30}{6}\right) = F(-0.84) \\ &\equiv \frac{x-30}{6} = -0.84 \\ &\equiv x = -0.84 * 6 + 30 \\ &\equiv x = 24.96 \end{aligned}$$

Pauta Pregunta 2:

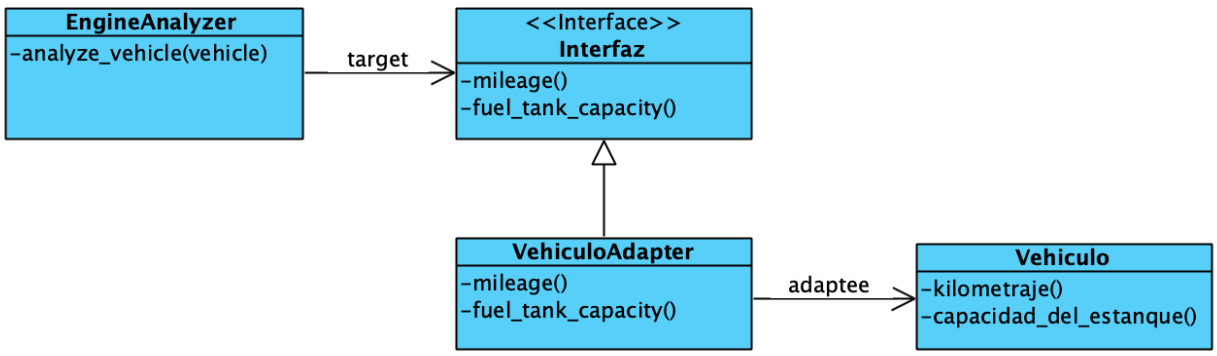
Inciso a):

```ruby
class VehiculoAdapter
  def initialize(vehiculo)
    @vehiculo = vehiculo
  end

  def mileage
    @vehiculo.kilometraje / 1.61
  end

  def fuel_tank_capacity
    @vehiculo.capacidad_del_estanque / 3.79
  end
end

analyzer = EngineAnalyzer.new
vehiculo = Vehiculo.new(10000, 50)
adapter = VehiculoAdapter.new(vehiculo)
analyzer.analyze_vehicle(adapter)
```

Nota de Pauta: En este ejemplo, se asume que `Vehiculo` tiene un constructor que acepta kilometraje y capacidad del estanque como parámetros. Se puede crear un `Vehiculo` con un constructor vacío. Se puede también asumir que `analyze_vehicle` es un método de clase y no de instancia.

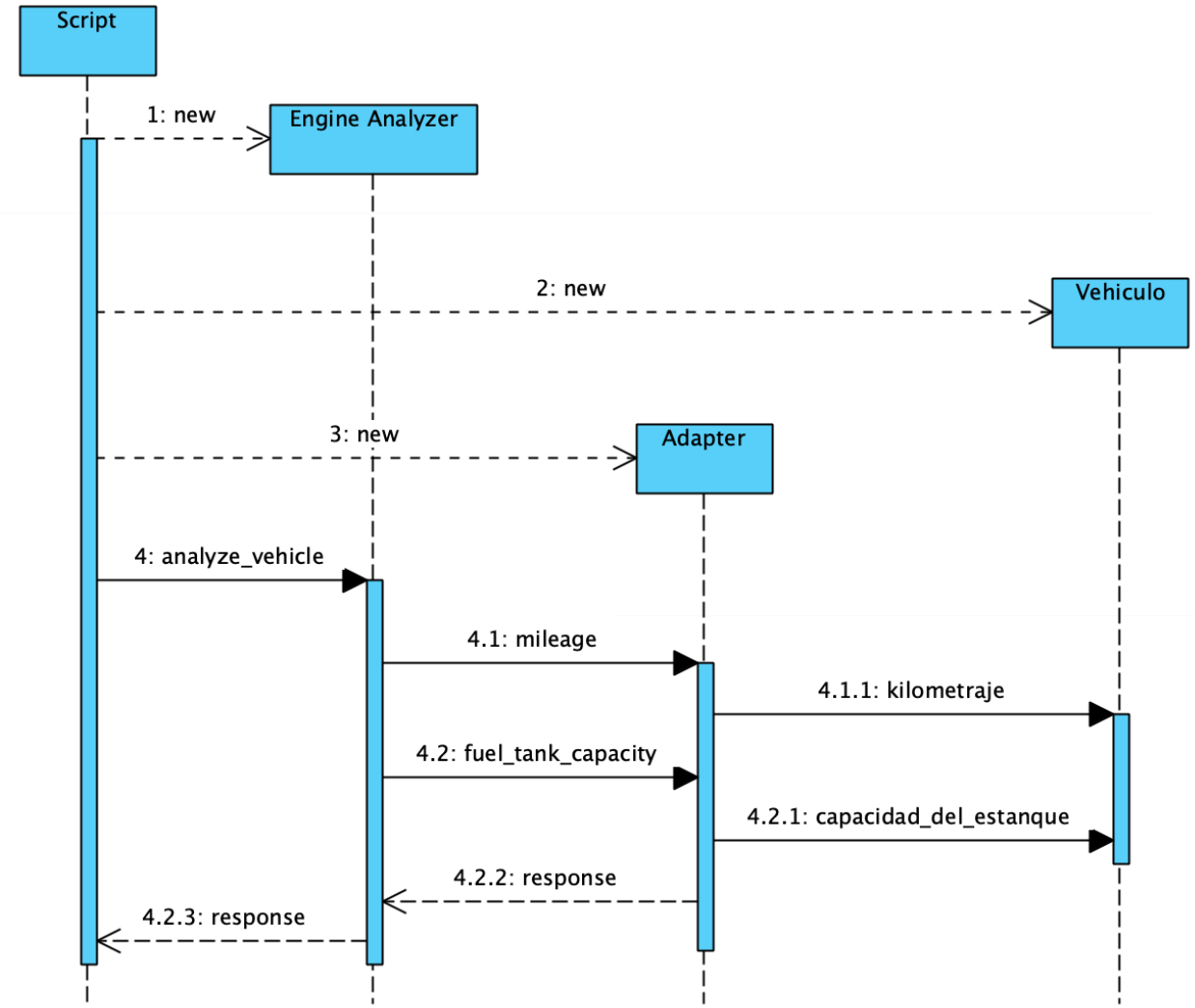
Ser criterioso en la evaluación del código: omitir errores sintácticos menores, solo descontar puntaje en caso de errores muy recurrentes o flagrantes. En caso de desarrollar este inciso en un lenguaje distinto de Ruby o en pseudocódigo, asignar 0 puntos.

Inciso b):



Nota de Pauta: Se espera que los alumnos especifiquen en el diagrama UML propuesto las operaciones de cada clase.

Inciso c):



Nota de Pauta: Es aceptable omitir la creación de los 3 lifelines distintos del script y asumir que están "vivos" en toda la secuencia. Así, se puede omitir la operación "new". Especificar respuestas de "kilometraje" y "capacidad_del_estanque" es opcional.

Pauta Pregunta 3:

Inciso a):

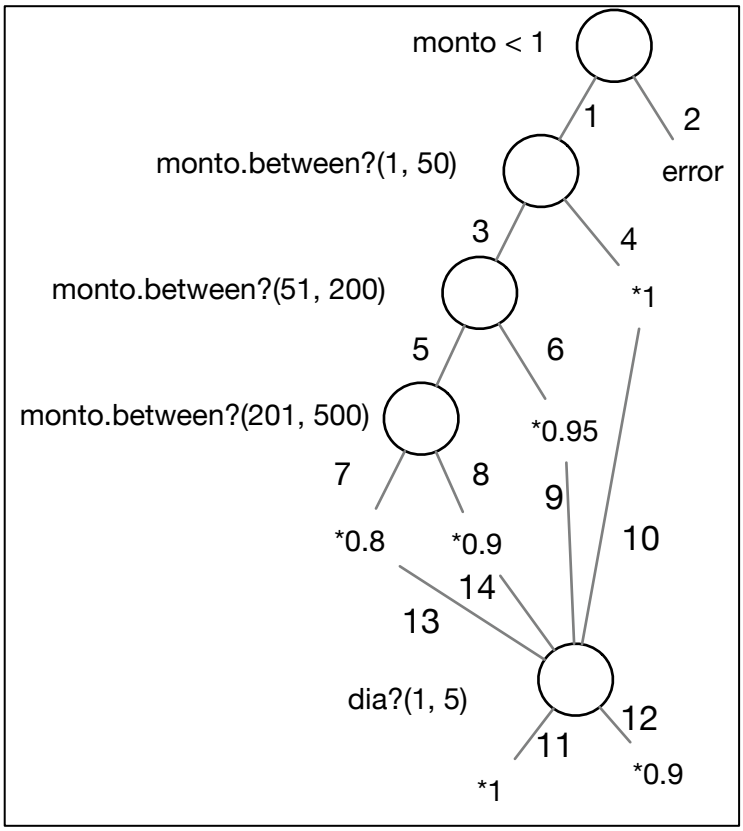
```ruby
def calcula(monto, dia)
  if monto < 1
    puts "el monto debe ser de al menos US$1"
    return
  elsif monto.between?(1, 50)
    nuevo_monto = monto
  elsif  monto.between?(51, 200)
    nuevo_monto = monto * 0.95
  elsif monto.between?(201, 500)
    nuevo_monto = monto * 0.90
  else nuevo_monto = monto * 0.80
  end
  if dia.between?(1,5)
    nuevo_monto *= 0.90
  else nuevo_monto
  end
end
```

Nota de Pauta: Se considera aceptable esta solución, donde se omiten valores inválidos de "día". Para valores inválidos, se espera que el alumno escriba un mensaje de consola, levante una excepción o puede retornar nil.

Ser criterioso en la evaluación del código: omitir errores sintácticos menores, solo descontar puntaje en caso de errores muy recurrentes o flagrantes. En caso de desarrollar este inciso en un lenguaje distinto de Ruby o en pseudocódigo, asignar 0 puntos.

Inciso b):

El siguiente grafo representa el código de la aplicación:



Posibles caminos y casos de prueba para cubrir cada uno de ellos:

Camino	monto	día
2	0	cualquiera
1 - 4 - 10 - 11	20	6
1 - 4 - 10 - 12	20	3
1 - 3 - 6 - 9 - 11	100	3
1 - 3 - 6 - 9 - 12	100	7
1 - 3- 5 - 8 -14 - 11	300	3
1 - 3- 5 - 8 -14 - 12	300	6
1 - 3- 5 - 7 -13 - 11	800	2
1 - 3- 5 - 7 -13 - 12	800	6

Con 9 casos de prueba se cubre el 100% de los caminos de ejecución.

Nota de Pauta: La solución propuesta permite conseguir path coverage para "all paths coverage". También se acepta que el alumno coloque menos tests siempre y cuando consiga "edge coverage" o "node coverage". En caso de considerar días inválidos en el inciso a), la solución del alumno debe contemplar cubrir ese camino adicional.

Inciso c):

Las clases de equivalencia quedan definidas por los intervalos de compra y los días de la compra

- Intervalos: (-∞, 1), [1, 50], (50, 200], (200, 500], (500, ∞)
- Días: [1,5], [6,7]

Ello define 10 clases de equivalencia (5 intervalos de monto y dos de días)
Un mínimo razonable de casos de prueba sería entonces 5 * 2 = 10 casos.

Adicionalmente, se pueden agregar los 8 casos de borde correspondientes a los 4 bordes (1, 50, 200, 500) para cada uno de los dos intervalos de días, con lo que llegamos a 10 + 8 = 18 casos
La tabla detalla la estrategia de testing para este problema.

Caso	monto	dia	resultado esperado
1	0	4	error
2	0	7	error
3	10	3	9
4	10	6	10
5	100	1	85.5
6	100	7	95
7	400	3	324
8	400	6	360
9	1000	5	720
10	1000	7	800
11	1	2	0.9
12	1	6	1
13	50	4	45
14	50	7	50
15	200	3	171
16	200	6	190
17	500	2	405
18	500	7	450

Nota de pauta: Para obtener todo el puntaje, basta con mencionar los 10 tests relacionados a clases de equivalencia. En caso de considerar días inválidos, la solución del alumno debe contemplar las clases de equivalencia:

- Días: [1,5], [6,7], (-∞,0], [8, ∞]

O bien:

- Días: [1,5], [6,7], (-∞,0] ∪ [8, ∞]

En cuyo caso debería definir 20 o 15 tests mínimos respectivamente.

Pauta Pregunta 4:

La planificación de un proyecto de software solía hacerse en forma detallada desde un principio. El resultado de ello era una *carta gantt* que describía el momento de inicio y de término de cada una de las actividades. Las *metodologías ágiles* como Scrum introducen la idea de una *planificación a nivel de sprint*, donde se detallan las tareas que se realizarán en un período corto y una *planificación a nivel de release* en que se planifica el número de iteraciones que existirán hasta el *release del producto*. A medida que transcurren *las iteraciones*, se vuelve a revisar el plan del release y puede que requiera cambios. Lo mas probable es que el trabajo a realizar se haya *subestimado*, por lo que ya no sería posible entregar en la fecha inicialmente planeada y haya que ajustar. El ajuste puede ser *por alcance* en que se reduce el número de features comprometidas dejando algunas menos prioritarias para *el siguiente release* o el ajuste puede ser *por tiempo*, en que debe renegociarse la fecha de entrega para acomodar una o dos iteraciones adicionales. El primero suele ser aceptado por el cliente (usuario) siempre que lo que quede fuera no sea crucial. Por ello es muy importante que quien priorice las funcionalidades que se implementan primero sea *el product owner* y no *el equipo de desarrollo*. Al no ajustar ni por alcance ni por tiempo lo que suele ocurrir es un *ajuste por calidad*, al intentar de cualquier forma hacer todo lo comprometido en la fecha original. Esta situación es *extremadamente riesgosa*, puesto que puede producirse una *deuda técnica* imposible de pagar o puede terminar entregándose un producto que no responderá a las expectativas o que presenta numerosas fallas.

En el caso de planificación de un sprint no hay posibilidad de ajustar *por tiempo* por lo que el ajuste se dará necesariamente *por alcance*. En este caso las funcionalidades planeadas no implementadas no necesariamente pasan a la siguiente iteración.

Para planificar es necesario hacer *estimaciones de esfuerzo* medido en *meses hombre* y de tiempo medido en *semanas o meses* de desarrollo. El input para este proceso de estimación es *el tamaño del software a construir* medido en líneas de código, *puntos de función o puntos de relato*. La ventaja de la primera métrica es que es fácil de registrar cuando el producto ya ha sido construido. La ventaja de los dos últimos es que no dependen del lenguaje o del estilo de programación utilizado. En el caso de los puntos de relato una técnica bastante utilizada es el llamado *scrum poker*, en que cada miembro de un grupo pequeño hace *una apuesta o predicción* sobre el puntaje de cada relato hasta llegar a un consenso. También es común que los puntajes que se asignen a los relatos desde muy simple a muy complejo no aumenten en forma lineal, sino en una escala exponencial o una *secuencia de fibonacci*.

Durante la época de desarrollo en cascada se elaboraron sofisticados modelos que relacionaban el tamaño con el esfuerzo o la duración (Cocomo) pero en realidad mas importante que la sofisticación del modelo es contar con *datos reales propios* de proyectos anteriores similares ojalá del mismo equipo de desarrollo.

Es una buena práctica hacer estimaciones de duración del proyecto que consideren tanto *una mirada optimista* como una *pesimista*, para exponer de manera explícita el *exceso de optimismo* de los desarrolladores. En efecto, el tiempo real suele estar mucho mas cerca del *mas largo* que del *mas corto*.

1. ajuste por alcance
2. ajuste por calidad
3. ajuste por comprensión
4. atraso en el proyecto
5. canasta de relatos
6. carta gantt
7. casos de uso
8. datos reales propios
9. datos validados de estudios internacionales
10. deuda técnica
11. el equipo de desarrollo
12. el jefe de proyecto
13. el product owner
14. el scrum master
15. el siguiente release
16. el siguiente sprint
17. el tamaño del software a construir
18. el tiempo necesario para desarrollar el producto
19. escala cuadrática
20. escala exponencial
21. estimaciones de duración
22. estimaciones de esfuerzo
23. estimaciones de tamaño
24. exceso de optimismo
25. exceso de pesimismo
26. extremadamente riesgosa
27. fin de la iteración
28. hoja de ruta
29. hombres mes (o meses hombre)
30. las iteraciones
31. líneas de código
32. mas corto
33. mas largo
34. mas probable
35. metodologías ágiles
36. muy positiva
37. número de funciones
38. pesimista (worst case)
39. planificación a nivel de portafolio
40. planificación a nivel de producto
41. planificación a nivel de release
42. planificación a nivel de sprint
43. por alcance
44. por alcance
45. por calidad
46. por calidad
47. por tiempo
48. por tiempo
49. puntos de función
50. puntos de relato
51. release del producto
52. scrum poker
53. secuencia de fibbonacci
54. semanas (o meses)
55. serie de taylor
56. sobreestimado
57. subestimado
58. tablero kanban
59. tablero kanban
60. tiempo
61. una apuesta o predicción
62. una mirada optimista (best case)

Nota de pauta: Para la corrección, considerar como equivalentes las opciones 1 (ajuste por alcance) y 43, 44 (por alcance); 2 (ajuste por calidad) y 45, 46 (por calidad); 47, 48 (por tiempo) y 60 (tiempo); a pesar de que gramáticamente hay una opción que calza mejor que otra. Para las partes donde se enumeran ítemes, el orden en que los haya puesto el alumno es irrelevante (e.g. puntos de función, puntos de relato, líneas de código; escala exponencial o secuencia de Fibonacci).

Hay 35 términos que el alumno debía reconocer. Descontar 0.2 puntos por cada error.