## 1 Utvecklingscykler

Para ihop följande begrepp med rätt utvecklingscykel
**Matcha ihop värdena:**

| | Kravinsamling | Analys | Design | Implementation | Testning |
|---|---|---|---|---|---|
| Interaktionsdiagram | | | ✔ | | |
| Systemsekvensdiagram | | ✔ | | | |
| User Stories | ✔ | | | | |
| Enhetstester (Unit tests) | | | | ✔ | |
| Use Cases | | ✔ | | | |
| Integrationstestning | | | | | ✔ |
| Klassdiagram | | | ✔ | | |

Totalpoäng: 7

## 2 Vattenfallsmodellen

Vattenfallsmodellen består (i ordning) av:
**Välj ett alternativ:**

- Kravinsamling, Analys, Testning, Implementation
- Kravinsamling, Implementation, Testning
- Kravinsamling, Analys, Design, Implementation
- Kravinsamling, Analys, Design, Implementation, Testning ✔
- Analys, Design, Implementation

Totalpoäng: 1

## 3 Planering

Markera de korrekta svaren nedan.

**Välj ett eller flera alternativ:**

- [ ] Ett GANTT-schema visar beroenden mellan olika arbetsuppgifter. ✔
- [ ] Ett GANTT-schema visar vem som skall göra vad i ett projekt. ✔
- [ ] I en WBS (Work Breakdown Structure) kan man direkt se när projektet förväntas vara färdigt.
- [ ] en WBS (Work Breakdown Structure) är en bild över vad ett system skall göra.

Totalpoäng: 2

## 4 SCRUM

Markera de påståenden som är sanna om SCRUM.
**Välj ett eller flera alternativ:**

- [ ] En backlog är en prioriterad lista med krav. ✔
- [ ] En SCRUM-cykel varar ungefär ett år.
- [ ] Varje SCRUM-cykel bör avslutas med att man kan leverera någontin ✔ ll kunden.
- [ ] SCRUM är en metod för att testa programvara.

Totalpoäng: 2

## 5 Interaktionsdiagram

Markera de korrekta svaren nedan:

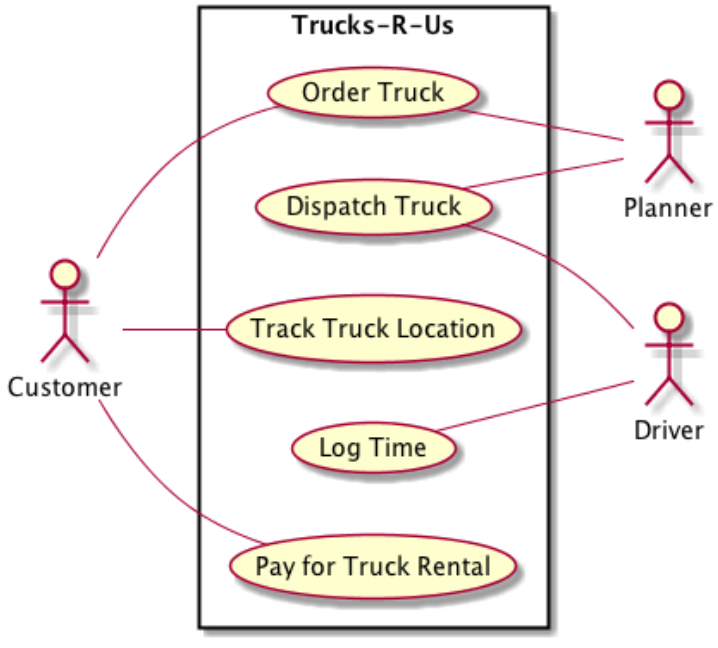
**Välj ett eller flera alternativ:**

- [ ] I sekvensdiagram listar man alla attribut och deras värden längst ner under varje objekts livlina.
- [ ] Alla objekt i ett interaktionsdiagram måste ha ett variablelnamn och en klasstyp, t.ex.: **namn:Typ**.
- [ ] Man gör ett interaktionsdiagram för varje systemhändelse. ✔
- [ ] Ett samarbetsdiagram beskriver samma sak som ett sekvensdiagrar. ✔
- [ ] Man gör ett systemsekvensdiagram för varje Use Case. ✔
- [ ] Man extraherar systemhändelser ur use-cases genom att rita ett systemsekvensdiagram. ✔
- [ ] Ett sekvensdiagram beskriver interaktionen mellan olika klasser.
- [ ] Syftet med systemsekvensdiagram är att identifiera hur olika aktörer interagerar med varandra.
- [ ] Man kan alltid använda tillståndsdiagram (State diagrams) i stället för interaktionsdiagram.

Totalpoäng: 4

## 6 Use Case Diagram

Betrakta Use-Case Diagrammet nedan. Markera sedan vilka alternativ som är korrekta.



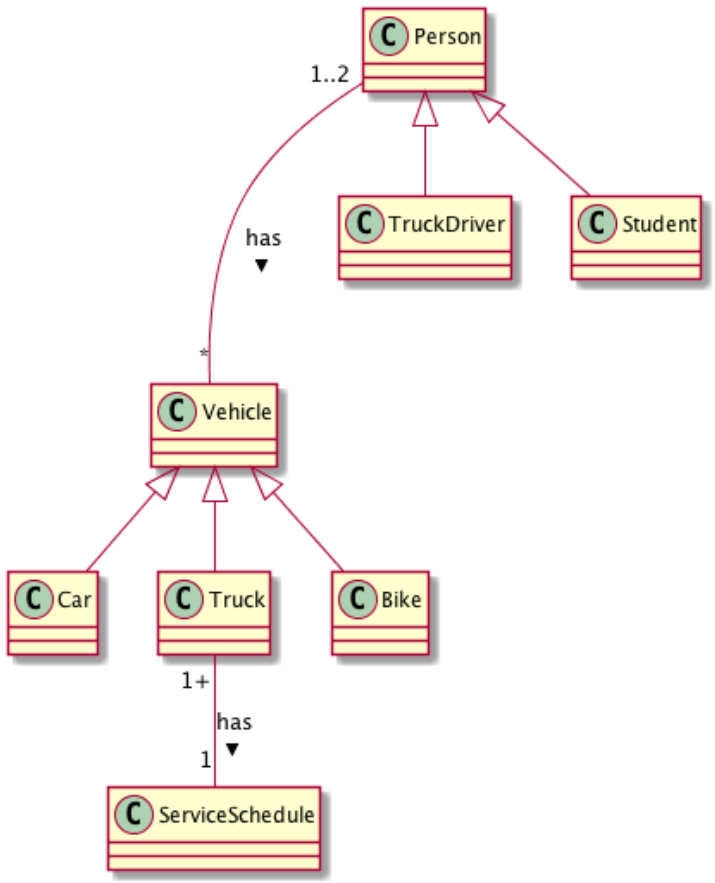
**Välj ett eller flera alternativ:**

- [ ] För att betala en lastbilshyra måste Föraren först ha loggat tidsåtgången i systemet. Det borde alltså finnas ett "includes"-förhållande mellan "Pay for Truck Rental" och "Log Time".
- [ ] För att kunna spåra en lastbil behöver man först ha beställt en lastbil.
- [ ] Kunden möter aldrig Föraren. ✔
- [ ] Planeraren interagerar med Föraren (via systemet) för att skicka iväg en lastbil. ✔

Totalpoäng: 2

## 7 Klassdiagram

Betrakta klassdiagrammet nedan. Markera sedan de påståenden som stöds av diagrammet.



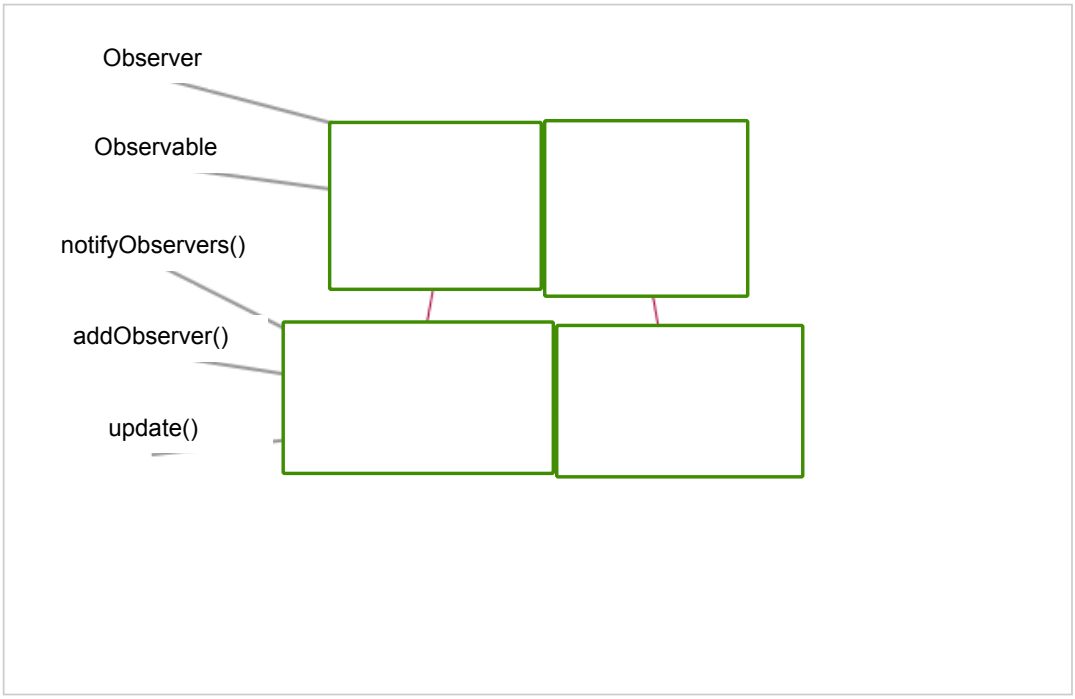
**Välj ett eller flera alternativ:**

- [ ] **Student**erna **Jack**, **Jill**, och **Jim** äger tillsammans en **Bike**.
- [ ] **Gyllenstierna**, som är en **Person**, äger **Jaguar:Car**, **Landrover:Ca** ✔ och **Volvo:Truck**.
- [ ] **Grållen** som är en **Car** ägs gemensamt av **Eva:Person**, ✔ och **Egon:TruckDriver**.
- [ ] **Tom**, som är en **Student**, äger **Skrothögen** som är en **Truck**. ✔
- [ ] Ett **ServiceSchedule** gäller bara för en **Truck** åt gången.
- [ ] Varje **Truck** har minst ett **ServiceSchedule**.
- [ ] Ett **Vehicle** måste vara antingen en **Car**, en **Truck**, eller en **Bike**.

Totalpoäng: 3

## 8 Observer Pattern

1. Drag och släpp klasserna **Observer** och **Observable** så att **OrderGenerator** och **WorkerNode** ärver från rätt klass.
2. Drag och släpp metoderna **notifyObservers()**, **addObserver()**, och **update()** till rätt klass (**OrderGenerator** eller **WorkerNode**).




Totalpoäng: 5

## 9 Design Patterns

I ett system finns en klass **NetworkUnitManager** som håller koll på en nätverksenhet. När någonting händer på nätverksenheten skall olika klasser

anropas. Till exempel skall klassen **LogEvent** logga att något hänt. **NetworkUnitViewer** skall visa att något hände. **PriceCalculator** skall ta reda på vad som hände och räkna upp priset för användning av enheten. Vi vill kunna lägga till dessa klasser, och utöka med fler liknande klasser utan att behöva ändra i **NetworkUnitManager** varje gång.

Vilket Design Pattern är bäst lämpat för detta beteende: (Builder, Observer, Strategy, State).

Totalpoäng: 1

## 10 Patterns

Markera de korrekta alternativen nedan.

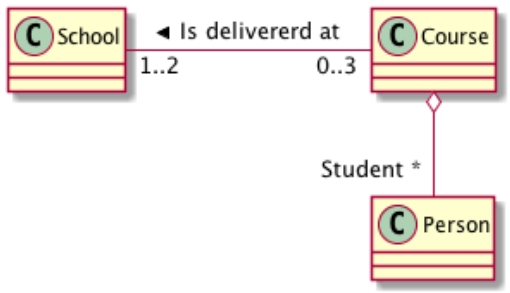
**Välj ett eller flera alternativ:**

- Om jag behöver ett kontrollerat sätt att radera objekt av en viss klass så använder jag mig av Proxy Pattern.
- I designmönstret State Pattern sparar man värdet på alla attribut (ett objekts tillstånd) i en separat klass.
- Model View Controller är ett arkitekturmönster för applikationer med olika sätt att visa en underliggande datamodell. ✔ a
- Det får bara finnas en instans av en Singleton-klass i ett system. ✔

Totalpoäng: 2

## 11 Relationer mellan Klasser

Betrakta klassdiagrammet nedan. Markera sedan de påståenden som stöds av diagrammet.



**Välj ett eller flera alternativ:**

- ☐ Det får högst finnas sex **Course**s i det här systemet.
- ☐ **K101:Course** undervisas vid **BTH:School** och **KTH:School** ✔
- ☐ **Molgan:Person** går inte några kurser. ✔
- ☐ **K404:Course** ges inte vid någon **School**.
- ☐ **K500:Course**, som ges vid **NF:School**, har inga **Student**er ✔
- ☐ **KTH:School** har två kurser, **A:Course** och **B:Course** ✔

Totalpoäng: 4

## 12 Domänmodell

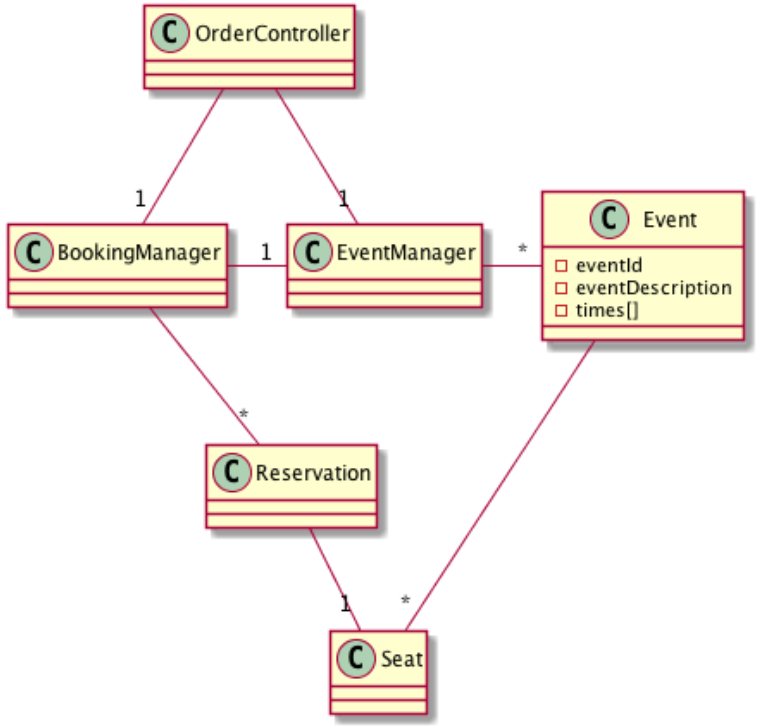
Markera de korrekta alternativen nedan:
**Välj ett eller flera alternativ:**

- ☐ Man använder sig av Domänmodellen när man börjar göra interaktionsdiagram. ✔
- ☐ En Domänmodell visar relationer mellan koncept. ✔
- ☐ En Domänmodell visar attribut på koncept. ✔
- ☐ En Domänmodell är ett förstadium till ett Klassdiagram. ✔
- ☐ I en Domänmodell betraktar man systemet som en "Black Box" och intresserar sig bara för det externa gränssnittet.
- ☐ En Domänmodell visar objekt och deras relationer.

Totalpoäng: 4

## 13 GRASP mönster

Betrakta först klassdiagrammet. Välj sedan vilka klasser som ansvarar för vad, och varför.




Klassen (EventManager, BookingManager, Event, Reservation, Seat, OrderController) är bäst lämpad att ta emot systemhändelsen **createBooking()** enligt GRASP mönstret (Information Expert, Creator, Controller, Low Coupling, High Cohesion).

Klassen (OrderController, BookingManager, EventManager, Event, Reservation, Seat)skall ha metoden **searchBooking()** eftersom den är en (Information Expert, Creator, Controller).

När en ny **Reservation** skall skapas så måste först (OrderController, BookingManager, Reservation, Seat) söka fram en **Seat** på ett **Event**. Denna klass får ansvaret enligt principen (Low Coupling, High Cohesion, Polymorphism, Pure Fabrication).

Totalpoäng: 6