***Opgave 3***

*Kun je een instantie maken van een interface via new? Leg uit waarom het logisch is dat het wel of niet kan.*

Nee. Er kan geen instantie gemaakt van een interface. Dat is logisch, omdat een interface eigenlijk gewoon een abstracte klas is zonder implementaties van methodes. De methodes zijn bepaald, maar je hebt eigenlijk alleen maar de signatuur en meer niet. Geen implementatie.

*Herhaal de vorige vraag met abstracte klassen.*

Een abstracte klasse werkt hetzelfde omdat deze geen complete implementatie hebben. Daarom kan je er geen instantie van maken.

*Kan een klasse meerdere klassen overerven?*

Nee, dit kan niet in Java. Hiervoor kan je interfaces gebruiken.

*Kan een klasse meerdere interfaces implementeren?*

Ja, dit kan wel.

*Kan een klasse tegelijk een klasse overerven en interfaces implementeren?*

Ja, maar je kan niet meer dan één klasse overerven – wel meerdere interfaces.

*Klopt de stelling dat elke methode in een interface abstract is? Licht je antwoord toe.*

Ja, elke methode in een interface is abstract. Een abstracte *klasse* kan wel gedefinieerde methodes hebben, maar een interface niet. Een interface is eigenlijk puur abstract, en heeft meer het doel dat het een soort “contract” omdat de methodes geïmplementeerd moeten worden in de interface-implementerende klasse.

*Moet een klasse abstract zijn als minstens één methode abstract is? Licht je antwoord toe.*

Ja, omdat je anders mogelijk de klasse zou instantiëren en mogelijk de abstracte methode aan zou roepen, die zonder implementatie zit. Dan heb je natuurlijk een probleem.

*Leg het begrip polymorfisme van klassen uit en geef twee voorbeelden (één met abstracte klassen en één met interfaces).*

Polymorfisme betekent dat een (of meerdere) subklasse de methode van een parentklasse aanpast. Polymorfisme betekent “veel vormen” en dat zie je dan terug: de methode heeft meerdere vormen in de childklassen.

**Opgave 4**

*a. Kan een klasse abstract zijn als geen enkele methode abstract is in die klasse? Probeer het eens uit. Leg waarom het logisch is dat dit wel of niet kan.*

Ja, het kan wel. Andersom kan niet. Het is logisch, omdat je van sommige klassen geen objecten wilt maken maar misschien wel de implementaties van de methodes mee wilt geven.

*b. Moet een subklasse van een abstracte klasse altijd alle abstracte methodes implementeren? Leg uit waarom het logisch is dat dit wel of niet kan.*

Ja, dit moet altijd. De hele bedoeling van een abstracte klasse (of een interface) is dat het eigenlijk een soort sjabloon geeft die bepaalt welke methodes er in je subklassen moeten zitten.

*c. Als een klasse niet alle methoden van een interface implementeert kun je iets doen om een (compiler)fout te voorkomen. Wat? Waarom is de oplossing logisch?*

Je kan je klasse abstract labelen zodat je niet alle methodes hoeft te implementeren. Dit is logisch omdat abstracte klassen niet alle methodes implementeren.

*d. Leg uit waarom het logisch is dat een instantie variabele niet abstract kan zijn.*

Een variabele hoef en kan je niet implementeren, dus je kan hem ook niet abstract labelen want er is helemaal geen implementatie om weg te laten.

*e. (Uitdaging) Zoek uit wat een final methode is. Leg daarna uit waarom het logisch is dat een methode niet tegelijkertijd abstract en final kan zijn.*

Een *final* methode is eigenlijk eht tegenovergestelde van een abstracte methode. Waar een abstracte methode juist overriden moet worden en geïmplementeerd moet worden in de subklassen, is final juist een methode die helemaal niet overriden kan worden en de implementatie kan niet aangepast worden. Het is dus eigenlijk het tegenovergestelde van een abstracte methode, en daarom kan een final methode nooit abstract zijn.