The following passage is adapted from an article, written by Michael Greshko in the June 2017 issue of *National Geographic* magazine. Parts 1 and 2 of this assessment, reference this passage.

Passage

On the afternoon of March 21, 2011, a heavy-equipment operator named Shawn Funk was carving his way through the earth, unaware that he would soon meet a dragon.

That Monday had started like any other at the Millennium Mine, a vast pit some 17 miles north of Fort McMurray, Alberta, operated by energy company Suncor. Hour after hour Funk's towering excavator gobbled its way down to sands laced with bitumen—the transmogrified remains of marine plants and creatures that lived and died more than 110 million years ago. It was the only ancient life he regularly saw. In 12 years of digging he had stumbled across fossilized wood and the occasional petrified tree stump, but never the remains of an animal—and certainly no dinosaurs.

Nearly six years later, I'm visiting the fossil prep lab at the Royal Tyrrell Museum in the windswept badlands of Alberta. The cavernous warehouse swells with the hum of ventilation and the buzz of technicians scraping rock from bone with needle-tipped tools resembling miniature jackhammers. But my focus rests on a 2,500-pound mass of stone in the corner. At first glance the reassembled gray blocks look like a nine-foot-long sculpture of a dinosaur. A bony mosaic of armor coats its neck and back, and gray circles outline individual scales. Its neck gracefully curves to the left, as if reaching toward some tasty plant. But this is no lifelike sculpture. It's an actual dinosaur, petrified from the snout to the hips.

For paleontologists the dinosaur's amazing level of fossilization—caused by its rapid undersea burial—is as rare as winning the lottery. Usually just the bones and teeth are preserved, and only rarely do minerals replace soft tissues before they rot away. There's also no guarantee that a fossil will keep its true-to-life shape. Feathered dinosaurs found in China, for example, were squished flat, and North America's "mummified" duck-billed dinosaurs, among the most complete ever found, look withered and sun dried. The remarkable fossil is a newfound species (and genus) of nodosaur, a type of ankylosaur often overshadowed by its cereal box–famous cousins in the subgroup Ankylosauridae

The western Canada that this dinosaur knew was a very different world from the brutally cold, windswept plains I encountered this past autumn. In the nodosaur's time, the area resembled today's South Florida, with warm, humid breezes wafting through conifer forests and fern-filled meadows. It's even possible that the nodosaur gazed out on an ocean. In the early Cretaceous, rising waters carved an inland seaway that blanketed much of what's now Alberta, its western shore lapping against eastern British Columbia, where the nodosaur may have lived. Today those ancient seabeds lie buried under forests and rolling fields of wheat.

Part 1 (22 marks): Classes, Objects, Information Hiding, Overloading

This section involves creating classes with state and behaviour that illustrate information hiding and overloading principles. All modelling is based on the passage on page 2.

(a) Examine the passage on page 2 and identify **SIX** core entities/concepts that are central for modelling a particular domain area in the passage. Explain why you chose each core entity/ concept and describe your domain area in 3-5 lines to provide a context for your work.
Tip: to identify a core concept in a domain area, search for key nouns used in descriptions.
[6 marks]

(b) Write working Java code for **THREE** classes that model specific aspects (state and behaviour) of three core entities/concepts from Part 1(a). Overall, your code must demonstrate the following:

- Information Hiding (FOUR different examples: TWO applied to state, TWO applied to behaviour)
- Overloading (TWO different examples applied to behaviour)

Note: Your classes are each required to have at most THREE state attributes (with different variable types), and at least TWO non-trivial methods that model relevant behaviour for a given class. Overloading examples cannot be based on accessors or mutators. [10 marks]

(c) Copy and fill out the table below, carefully explaining how your work satisfies the criteria in Part 1 (b). [6 marks]

Criteria	Class Name	Specific aspect of the class	Object-Oriented construct(s) used	How is the criteria satisfied?
Information Hiding (example 1)				
Information Hiding (example 2)				
Information Hiding (example 3)				
Information Hiding (example 4)				
Overloading (example 1)				
Overloading (example 2)				

End of Part 1
Please turn to the next page

Part 2 (26 marks): Relationships: Association, Aggregation and Composition

This section involves further developing the classes you have created in Part 1, by setting up relationships between the classes.

(a) In the classes created in Part 1, write working Java code to model **TWO** examples each of the following relationships:

- Association
- Aggregation

Note: All relationships must be derived from or inspired by the passage. You must also describe the purpose of each relationship briefly. [8 marks]

(b) Copy and fill out the table below, carefully explaining how your work satisfies the criteria in Part 2 (a). [6 marks]

Criteria	Participating Classes (names)	Relationship Cardinality	Object-Oriented construct(s) used	How is the criteria satisfied?
Association (example 1)				
Association (example 2)				
Aggregation (example 1)				
Aggregation (example 2)				

(c) Draw a simple UML diagram that illustrates the relationships between your classes. Include relationship names and cardinalities on all relationship lines. Correct UML notation is required. [6 marks]

(d) Write working Java code to create TWO instances of each your classes and set up the relationships created in Part 2 (a). If an instance requires data, explain how that data is passed in. If no data is required, explain why you designed your classes this way. [6 marks]

End of Part 2
Please turn to the next page

Part 3 (20 marks): Inheritance and Polymorphism.

Consider the following class signature:

```
public final class Integer extends Number implements Comparable<Integer>{ }
```

Determine whether the following statements will compile and/or run. Explain completely in your own words why the statements will compile and/or run, referencing the appropriate subclasses, superclasses, subtypes or supertypes involved. Identify the output or errors produced and clearly describe the full reason(s) for any output produced.

The marks for the questions in this section are not equally distributed.

(i)
```
Integer one = new Number(1);
System.out.println(one);
```

(ii)
```
Number two = new Integer("two");
System.out.println(two);
```

(iii)
```
Integer three = new Comparable();
System.out.println(three);
```

(iv)
```
java.lang.Comparable<> four = new Integer("4");
System.out.println(four);
```

(v)
```
Integer five = new Integer(5);
System.out.println(five);
```

(vi)
```
java.lang.Comparable six = new Integer("this is a 6");
System.out.println(six);
```

(vii)
```
Number seven = Integer.valueOf(7);
System.out.println(seven);
```

(viii)
```
Number eight = (Object)Integer.parseInt("8");
System.out.println(eight);
```

(ix)
```
Object nine = new Object();
System.out.println(nine);
```

(x)
```
Integer ten = new Integer("ten".compareTo("eleven"));
System.out.println(ten);
```

End of Part 3
Please turn to the next page