CS FINAL Code Reference
2019 B

```
public class Creature {

    private double weight;

    public Creature(double weight){
        this.weight=weight;
        openEyes();
    }

    public void openEyes() {
        System.out.println("Creature eyes opening");
    }

    //other methods not shown

}

public class LandCreature extends Creature{

    private int legCount;

    //construtor not shown

    public void openEyes(){
        System.out.println("LandCreature eyes opening");
    }
}

public class Vehicle {

    private int hp;
    private int wheelCount;

    public Vehicle(int hp, int wheelCount){
        this.hp=hp;
        this.wheelCount=wheelCount;
        startEngine();
    }

    public void startEngine() {
        System.out.println("rumble rumble rumble");
    }
}

public class DumpTruck extends Vehicle {

    private int maxCargoWeight;

    //constructor not shown, doesn't call any non-constructor methods

    public void startEngine() {
          System.out.println("glug glug glug glug glug");
    }

}
```

```
public class Car {

    private String licensePlate;
    private int position;
    private int speed;

    public Car() {
        System.out.println("Car created!");
    }

    public Car(String licensePlate, int position, int speed) {
        System.out.println("Car created with overloaded constructor!");
        this.licensePlate = licensePlate;        this.position = position;        this.speed = speed;
    }

    public void printSpeed() {
        System.out.println("Speed is " + speed);
    }

    public void speedUp()
        { speed++;
    }

    public void speedUp(int amount) {
        speed+=amount;
        if (speed>200)
            speed=200;
    }

    public void slowDown(int amount) {
        speed-=amount;
        if (speed < 0 )
            speed = 0;
    }

    public void slowDown()
        { speed--;
        if (speed < 0 )
            speed = 0;
    }

    public int getSpeed() {
        return(speed);
    }

    public int getPosition() {
        return(position);
    }

    !!!  Car class continued on next
         page  !!!
    public String getLicensePlate() {
        return(licensePlate);
    }

    public void honk(int count) { for(int k=0; k<count; k++)
            System.out.println("beep");
    }

    public String toString(){
        return("Car: " + licensePlate + ", position is " + position + ", speed is " + speed);
    }

}
```

```
public class Climber {

    public String name;
    private int age;
    private int altitude;

    public Climber() {
        name="NoName"; age=-1; altitude=0;
        showInfo();
    }

    public Climber(String name) {
        this.name = name; age=-1; altitude=0;
        showInfo();
    }

    public Climber(String name, int age, int altitude) {
        this.name=name; this.age = age; this.altitude=altitude;
        showInfo();
    }

    public void gainAltitude(int distance) {
        altitude+=distance;
    }

    public int getAltitude() {
        return(altitude);
    }

    public int distanceRemaining(int height) {
        int diff = height - altitude;
        if (diff>=0)
            return(diff);
        else
            return(0);
    }

    public int altitudeDifference(int alt1, int alt2){
        int diff = alt2 - alt1;
        return(diff);
    }

    public String getName() {
        return(name);
    }

    public String toString() {
        return("Climber named " + name + " at " + altitude + " meters.");
    }

    public void showInfo() {
        System.out.println("Climber Info: ");       System.out.println("Name is: " + name);
        System.out.println("Age is: " + age);       System.out.println("Altitude is: " + altitude);
    }
}
```

```
public class Shape {

    private double x,y;
    public int sides;

    public Shape()
        { x=0; y=0;
    }

    public Shape(double x, double y) {
        this.x=x; this.y=y;
    }

    public double getX() {
        return(x);
    }
    public double getY()
        { return(y);
    }
    public void setX(double x)
        { this.x=x;
    }
    public void setY(double y)
        { this.y=y;
    }
    public int getSides() {
        return(sides);
    }
    public void move(int xdistance, int ydistance) {
        x+=xdistance;
        y+=ydistance;
    }
    public double getArea() {
        //no code yet... left later for abstract method...
        return(0);
    }
    public double getPerimeter() {
        //no code yet... left later for abstract method...
        return(0);
    }
    public void draw(Graphics g) {
        //no code here yet... left later for abstract method...
    }

}
```

```java
public class Circle extends Shape {

    public double radius;

    public Circle(double radius){
        super();
        this.radius=radius;
    }

    public Circle(int x, int y, double radius){
        super(x,y);
        sides=1;
        this.radius=radius;
    }

    public double getArea(){
        return(2*Math.PI*radius*radius);
    }
    public double getPerimeter(){
        return(2*Math.PI*radius);
    }
    public void draw(Graphics g) {
        //code to draw the circle at its x,y location
    }

    private void shrink() {
        //code to shrink the circle in size
    }

}
public class ColoredCircle extends Circle {

    public Color myColor;

    public ColoredCircle(double radius, Color color){
        super(radius);
        myColor=color;
    }
    public ColoredCircle(int x, int y, double radius, Color
        color){
        super(x,y,radius);
        myColor=color;
    }

    public void draw(Graphics g){
        //code to draw this circle in color at x,y
    }
    public void drawWithoutColor(Graphics g){
        super.draw(g);
    }
}
```

```
public class Person {

    public String name="N/A";
    public Address address;
    private int age;
    protected double weight;

    public Person(String name, int age){
        this.name=name; this.age=age;
    }

    public void talk() {
        System.out.println("Person Talk...");
    }

    public void breath() {
        System.out.println("Breathing...");
    }

    public int getAge(){
        return(age);
    }

    public void setAge(int age){
        this.age=age;
    }

    protected void stuff() {
       System.out.println("Stuff");
    }
}
```

```
public class Student extends Person {

    public int studentId;
    public ArrayList<Mark> marks=new ArrayList<Mark>();

    public Student(String name, int age, int studentId) {
        super(name, age);
        this.studentId = studentId;
    }

    public void study(){
        System.out.println("studying...");
    }

    public void talk() {
        System.out.println("Student Talk...");
    }

    public double getAverageMark(){
        if (marks.isEmpty())
            return(0);
        double sum=0;
        for(int k=0; k<marks.size(); k++)
            sum+=marks.get(k).percent;
        return(sum/marks.size());
    }

    public void partnerUp(Student S){
        //code not shown
    }

}
```

```
public class ExchangeStudent extends Student {

    public ExchangeStudent(String name, int age, int studentId) {
        super(name, age, studentId);
        System.out.println("Created ExchangeStudent");
    }


    public void talk() {
        System.out.println("ExchangeStudent Talk...");
    }

    //doesn't include English mark
    public double getAverageMark() {
        double sum=0;
        int courseCount=0;
        for(int k=0; k<marks.size(); k++)
            if (marks.get(k).courseName.equals("English")==false)
                { sum+=marks.get(k).percent;
                courseCount++;
            }

        if (courseCount==0)
            return(0);
        return(sum/courseCount);
    }

    //returns the same average mark that would be calculated for a regular
    Student public double getUnadjustedAverageMark() {
        double avg =
        super.getAverageMark(); return(avg);
    }

    public String getReturnDate() {
        //returns the date they are returning home - code that looks up in
        database return("June 30, 2020");
    }

}
```

```
public class Location {

    private String city;
    private String country;

    public Location() {
       city="not provided";
       country="not provided";
    }

    public Location(String city, String
       country){
       this.city = city;
       this.country = country;
        }

    public String getCity(){
       return(city);
    }

    public String getCountry() {
       return(country);
    }

    public void setCity(String city){
       this.city = city;
    }

    public void setCountry(String
       country){
       this.country=country;
        }

}
```

```
public class LuckyNumbers {

  public int[] numbers;
  private int count;

  public LuckyNumbers(){
    //for testing purposes we will randomly fill this instance with
    numbers count = 3;
    numbers = new int[count];
    for(int k=0; k<numbers.length; k++)
      numbers[k] = (int)(Math.random()*1000);
  }

  public int getLuckyNumber(int index){
    if ( (index<0) ||
      (index>=numbers.length)) return(0);
    return(numbers[index]);
  }

  public int getCount(){
   return(count);
  }

  public void addLuckyNumber(int x){
    count++;
    int[] temp = new int[count];
    for(int k=0; k<number.length; k++){
       temp[k] = numbers[k];
       temp[temp.length-1]=x;
    }
    numbers=temp;

}
```