GuessMaster.java:

```
package PersonAndDate;
import java.util.Scanner;
import java.util.Random;


public class GuessMaster {
    private int numMaxEntities;
    private Entity[] entities;
    private int totalTickets;

    //Constructor
    public GuessMaster() {
        this.numMaxEntities = 0;
        this.entities = new Entity[100];
        this.totalTickets = 0;
    }

    //Method to add cloned entity
    public void addEntity(Entity entity) {
        if (numMaxEntities < 100) {
            entities[numMaxEntities] = entity.clone();
            numMaxEntities++;
        } else {
            System.out.println("Error: Maximum entities reached.");
        }
    }

    //Generating random entity
    private int genRandomEntityInd() {
        Random rand = new Random();
        return rand.nextInt(numMaxEntities);
    }

    //Playing GuessMaster using specific entity
    public void playGame(Entity entity) {
        Scanner scanner = new Scanner(System.in);
        System.out.println("***************************");
        System.out.println(entity.welcomeMessage());
        System.out.println("Guess " + entity.getName() + "'s birthday (format:
mm/dd/yyyy)");

        while (true) {
            String userInput = scanner.nextLine().trim();

            //Checking for exit command
            if (userInput.equalsIgnoreCase("quit") ||
userInput.equalsIgnoreCase("exit")) {
                System.out.println("Thanks for playing! Exiting...");
                System.exit(0);
            }

            //Parse the users input as a date
            try {
```

```
                Date userDate = new Date(userInput);
                if (userDate.equals(entity.getBorn())) {
                    int awardedTickets = entity.getAwardedTicketNumber();
                    totalTickets += awardedTickets;
                    System.out.println("************* Bingo! ***************");
                    System.out.println("You won " + awardedTickets + " tickets in
this round.");
                    System.out.println("The total number of your tickets is " +
totalTickets + ".");
                    System.out.println(entity.closingMessage());
                    break;
                } else if (userDate.precedes(entity.getBorn())) {
                    System.out.println("Incorrect. Try a later date.");
                } else {
                    System.out.println("Incorrect. Try an earlier date.");
                }
            } catch (Exception e) {
                System.out.println("Invalid date format. Please try again.");
            }
        }
    }

    //Playing game for a specific entity index
    public void playGame(int entityInd) {
        if (entityInd >= 0 && entityInd < numMaxEntities) {
            playGame(entities[entityInd]);
        } else {
            System.out.println("Invalid entity index.");
        }
    }

    //Playing game with a randomly selected entity
    public void playGame() {
        if (numMaxEntities == 0) {
            System.out.println("No entities available to play.");
            return;
        }

        while (true) {
            int randomIndex = genRandomEntityInd();
            playGame(randomIndex);
        }
    }

    //Main method to initialize and start game
    public static void main(String[] args) {
        System.out.println("======================");
        System.out.println("GuessMaster 2.0");
        System.out.println("======================");
        System.out.println("Type 'quit' or 'exit' to end the game.");

        //Initialize entities

        Politician churchill = new Politician("Winston Churchill", new
Date("November", 30, 1874), "Male", "Conservative", 0.25);
```

```
        Singer dion = new Singer("Celine Dion", new Date("March", 30, 1961),
"Female", "La voix du bon Dieu", new Date("November", 6, 1981), 0.5);
        Person myCreator = new Person("myCreator", new Date("September", 1, 2000),
"Female", 1.0);
        Country usa = new Country("United States", new Date("July", 4, 1776),
"Washington D.C.", 0.1);

        GuessMaster gm = new GuessMaster();
        gm.addEntity(churchill);
        gm.addEntity(dion);
        gm.addEntity(myCreator);
        gm.addEntity(usa);

        gm.playGame();
    }
}
```

Entity.java

```
package PersonAndDate;

public abstract class Entity implements Cloneable {
    private String name;
    private Date born;
    private double difficulty;

    //Constructor
    public Entity(String name, Date born, double difficulty) {
        this.name = name;
        this.born = new Date(born);
        this.difficulty = difficulty;
    }

    //Copy Constructor
    public Entity(Entity other) {
        if (other == null) {
            System.out.println("Fatal Error");
            System.exit(0);
        }
        this.name = other.name;
        this.born = new Date(other.born);
        this.difficulty = other.difficulty;
    }

    //Accessor methods
    public String getName() {
        return name;
    }

    public Date getBorn() {
        return new Date(born);
    }

    public double getDifficulty() {
        return difficulty;
    }

    //Mutator methods
    public void setName(String name) {
        this.name = name;
    }

    public void setBorn(Date born) {
        this.born = new Date(born);
    }

    public void setDifficulty(double difficulty) {
        if (difficulty < 0 || difficulty > 1) {
            System.out.println("Invalid difficulty level. It must be between 0 and
1.");
            return;
        }
```

```
        this.difficulty = difficulty;
    }

    // Abstract Methods
    public abstract String entityType();
    public abstract Entity clone();

    //Ticket calculation
    public int getAwardedTicketNumber() {
        return (int) (difficulty * 100);
    }

    //Welcome and Closing Messages
    public String welcomeMessage() {
        return "Welcome! Let’s start the game! This entity is a " + entityType() +
"!";
    }

    public String closingMessage() {
        return "Congratulations! The detailed information of the entity you guessed
is: " + toString();
    }

    //toString method
    public String toString() {
        return name + ", born on " + born.toString();
    }

    //Equals method
    public boolean equals(Object obj) {
        if (this == obj) return true;
        if (obj == null || getClass() != obj.getClass()) return false;
        Entity entity = (Entity) obj;
        return name.equals(entity.name) && born.equals(entity.born);
    }
}
```

Date.java

```
package PersonAndDate;
import java.util.Scanner;

public class Date
{
    private String month;
    private int day;
    private int year;

    //Default constructor to initialize at 01/01/1000
    public Date( ){
        month = "January";
        day = 1;
        year = 1000;
    }

    //Constructor accepting month as String
    public Date(String monthString, int day, int year){
        setDate(monthString, day, year);
    }

    //Constructor accepting month as integer
    public Date(int monthInt, int day, int year){
        setDate(monthInt, day, year);
    }

    //Constructor that only takes year, defaults to January 1st
    public Date(int year){
        setDate(1, 1, year);
    }

    //Copy constructor for copy of another Date object
    public Date(Date aDate){
        if (aDate == null)
        {
             System.out.println("Fatal Error.");
             System.exit(0);
        }

        month = aDate.month;
        day = aDate.day;
        year = aDate.year;
    }

    //Constructor to parse string in mm/dd/yyyy format
    public Date(String strDate) {
        String[] parts = strDate.split("/");
        if (parts.length != 3 || !isNumeric(parts[0]) || !isNumeric(parts[1]) ||
!isNumeric(parts[2])) {
            System.out.println("Fatal Error");
            System.exit(0);
        }
        int month = Integer.parseInt(parts[0]);
```

```
    int day = Integer.parseInt(parts[1]);
    int year = Integer.parseInt(parts[2]);
    setDate(month, day, year);
}

//Method to check for only digits
private static boolean isNumeric(String str) {
    for (char c : str.toCharArray()) {
        if (!Character.isDigit(c)) {
            return false;
        }
    }
    return true;
}

//Mutator method to initialize integer month
public void setDate(int monthInt, int day, int year){
    if (dateOK(monthInt, day, year)){
        this.month = monthString(monthInt);
        this.day = day;
        this.year = year;
    }
    else{
        System.out.println("Fatal Error");
        System.exit(0);
    }
}

//Mutator method to initialize string month
public void setDate(String monthString, int day, int year)
{
    if (dateOK(monthString, day, year)){
        this.month = monthString;
        this.day = day;
        this.year = year;
    }
    else{
        System.out.println("Fatal Error");
        System.exit(0);
    }
}

//Mutator to update year only
public void setDate(int year){
    setDate(1, 1, year);
}

public void setYear(int year){
    if ( (year < 1000) || (year > 9999) ){
        System.out.println("Fatal Error");
        System.exit(0);
    }
    else
        this.year = year;
}
```

```
public void setMonth(int monthNumber)
{
    if ((monthNumber <= 0) || (monthNumber > 12))
    {
        System.out.println("Fatal Error");
        System.exit(0);
    }
    else
        month = monthString(monthNumber);
}

public void setDay(int day)
{
    if ((day <= 0) || (day > 31))
    {
        System.out.println("Fatal Error");
        System.exit(0);
    }
    else
        this.day = day;
}

public int getMonth( )
{
    if (month.equals("January"))
        return 1;
    else if (month.equals("February"))
        return 2;
    else if (month.equalsIgnoreCase("March"))
        return 3;
    else if (month.equalsIgnoreCase("April"))
        return 4;
    else if (month.equalsIgnoreCase("May"))
        return 5;
    else if (month.equals("June"))
        return 6;
    else if (month.equalsIgnoreCase("July"))
        return 7;
    else if (month.equalsIgnoreCase("August"))
        return 8;
    else if (month.equalsIgnoreCase("September"))
        return 9;
    else if (month.equalsIgnoreCase("October"))
        return 10;
    else if (month.equals("November"))
        return 11;
    else if (month.equals("December"))
        return 12;
    else
    {
        System.out.println("Fatal Error");
        System.exit(0);
        return 0; //Needed to keep the compiler happy
    }
}
```

```
public int getDay( )
{
    return day;
}

public int getYear( )
{
    return year;
}

public String toString( )
{
    return (month + " " + day + ", " + year);
    //value will be shown when debugging
    //a very useful feature for dubugging
    //also useful in println(), which
    //automatically call toString();
}

public boolean equals(Date otherDate) {
    return ( (getMonth() == otherDate.getMonth()) // Compare numerical values
              && (day == otherDate.day) && (year == otherDate.year) );
}


public boolean precedes(Date otherDate)
{
    return ( (year < otherDate.year) ||
       (year == otherDate.year && getMonth( ) < otherDate.getMonth( )) ||
       (year == otherDate.year && month.equals(otherDate.month)
                                    && day < otherDate.day) );
}

//Method to for user to input date through console (Not used)
public void readInput( )
{
    boolean tryAgain = true;
    Scanner keyboard = new Scanner(System.in);
    while (tryAgain)
    {
        System.out.println("Enter month, day, and year.");
          System.out.println("Do not use a comma.");
        String monthInput = keyboard.next( );
        int dayInput = keyboard.nextInt( );
        int yearInput = keyboard.nextInt( );
        if (dateOK(monthInput, dayInput, yearInput) )
        {
            setDate(monthInput, dayInput, yearInput);
            tryAgain = false;
        }
        else
            System.out.println("Illegal date. Reenter input.");
     }
}
```

```
//Ensuring integer month date fits calendar format
  private boolean dateOK(int monthInt, int dayInt, int yearInt)
  {
      return ( (monthInt >= 1) && (monthInt <= 12) &&
               (dayInt >= 1) && (dayInt <= 31) &&
               (yearInt >= 1000) && (yearInt <= 9999) );
  }

  //Ensuring string month date fits calendar format
  private boolean dateOK(String monthString, int dayInt, int yearInt)
  {
      return ( monthOK(monthString) &&
               (dayInt >= 1) && (dayInt <= 31) &&
               (yearInt >= 1000) && (yearInt <= 9999) );
  }

  //Ensures month string is written correctly
  private boolean monthOK(String month)
  {
      return (month.equals("January") || month.equals("February") ||
              month.equals("March") || month.equals("April") ||
              month.equals("May") || month.equals("June") ||
              month.equals("July") || month.equals("August") ||
              month.equals("September") || month.equals("October") ||
              month.equals("November") || month.equals("December") );
  }

  //Converting month integer to string
  private String monthString(int monthNumber)
  {
      switch (monthNumber)
      {
      case 1:
          return "January";
      case 2:
          return "February";
      case 3:
          return "March";
      case 4:
          return "April";
      case 5:
          return "May";
      case 6:
          return "June";
      case 7:
          return "July";
      case 8:
          return "August";
      case 9:
          return "September";
      case 10:
          return "October";
      case 11:
          return "November";
```

```
            case 12:
                return "December";
            default:
                System.out.println("Fatal Error");
                System.exit(0);
                return "Error"; //to keep the compiler happy
            }
        }
    }
```

Person.java

```
package PersonAndDate;

public class Person extends Entity {
    private String gender;

    //Constructor
    public Person(String name, Date born, String gender, double difficulty) {
        super(name, born, difficulty);
        this.gender = gender;
    }

    //Copy Constructor
    public Person(Person other) {
        super(other);
        this.gender = other.gender;
    }

    //Accessor method
    public String getGender() {
        return gender;
    }

    //Mutator method
    public void setGender(String gender) {
        this.gender = gender;
    }

    //Abstract method implementation
    public String entityType() {
        return "This entity is a person!";
    }

    //Clone method implementation
    public Person clone() {
        return new Person(this);
    }

    //Override toString method
    public String toString() {
        return "\nName: " + getName() + "\nBorn at: " + getBorn().toString() +
"\nGender: " + gender;
    }
}
```

Country.java

```java
package PersonAndDate;

public class Country extends Entity {
    private String capital;

    //Constructor
    public Country(String name, Date born, String capital, double difficulty) {
        super(name, born, difficulty);
        this.capital = capital;
    }

    //Copy Constructor
    public Country(Country other) {
        super(other);
        this.capital = other.capital;
    }

    //Acessor method
    public String getCapital() {
        return capital;
    }

    //Mutator method
    public void setCapital(String capital) {
        this.capital = capital;
    }

    //abstract method
    public String entityType() {
        return "country";
    }

    //Clone method
    public Country clone() {
        return new Country(this);
    }

    //Overriding toString method
    public String toString() {
        return super.toString() + ", Capital: " + capital;
    }
}
```

Politician.java

```
package PersonAndDate;

public class Politician extends Person {
    private String party;

    //Constructor
    public Politician(String name, Date born, String gender, String party, double
difficulty) {
        super(name, born, gender, difficulty);
        this.party = party;
    }

    //Copy Constructor
    public Politician(Politician other) {
        super(other);
        this.party = other.party;
    }

    //Accessor method
    public String getParty() {
        return party;
    }

    //Mutator method
    public void setParty(String party) {
        this.party = party;
    }

    // Implement abstract method entityType
    public String entityType() {
        return "This entity is a politician!";
    }

    //Clone method
    public Politician clone() {
        return new Politician(this);
    }

    //Overriding toString method
    public String toString() {
        return "\nName: " + super.getName() + "\nGender: " + getGender() + "\nParty:
" + party;
    }
}
```

Singer.java

```
package PersonAndDate;

public class Singer extends Person {
    private String debutAlbum;
    private Date debutAlbumReleaseDate;

    //Constructor
    public Singer(String name, Date born, String gender, String debutAlbum, Date
debutAlbumReleaseDate, double difficulty) {
        super(name, born, gender, difficulty);
        this.debutAlbum = debutAlbum;
        this.debutAlbumReleaseDate = new Date(debutAlbumReleaseDate);
    }

    //Copy Constructor
    public Singer(Singer other) {
        super(other);
        this.debutAlbum = other.debutAlbum;
        this.debutAlbumReleaseDate = new Date(other.debutAlbumReleaseDate);
    }

    //Accessor methods
    public String getDebutAlbum() {
        return debutAlbum;
    }
    public Date getDebutAlbumReleaseDate() {
        return new Date(debutAlbumReleaseDate);
    }

    //Mutator methods
    public void setDebutAlbum(String debutAlbum) {
        this.debutAlbum = debutAlbum;
    }

    public void setDebutAlbumReleaseDate(Date debutAlbumReleaseDate) {
        this.debutAlbumReleaseDate = new Date(debutAlbumReleaseDate);
    }

    //Abstract method implementation
    public String entityType() {
        return "This entity is a singer!";
    }

    public Singer clone() {
        return new Singer(this);
    }

    //Overriding toString method
    public String toString() {
        return super.toString() + ", \nDebut Album: " + debutAlbum + ", \nReleased
on: " + debutAlbumReleaseDate.toString();
    }
}
```