Python code to:

1) Tokenise the text
2) Perform sentiment analysis to produce sentiment scores separately for the whole novel as well as a selection of sentences relevant to two novel characters – Boayu and Daiyu.

```
import re
from snownlp import SnowNLP

def tokenize_chinese_text(text):
    """Tokenizes Chinese text into sentences."""
    sentence_endings = ["。", "! ", "? "]
    sentences = re.split("|".join(sentence_endings), text)
    return [sentence.strip() for sentence in sentences if sentence]

def extract_sentences_with_character(sentences, character_name):
    """Extracts sentences that mention a specific character."""
    return [sentence for sentence in sentences if character_name in sentence]

def calculate_sentiment(sentence):
    """Calculates sentiment score of a sentence using SnowNLP."""
    if not sentence.strip():  # check if sentence is empty or just whitespace
        return None
    try:
        return SnowNLP(sentence).sentiments
    except ZeroDivisionError:
        return None

def save_sentiments_to_file(filename, sentiments_by_chapter):
    """Saves sentiments to a file with chapter delineation."""
    with open(filename, "w", encoding="utf-8") as out:
        for chapter_num, sentiments in enumerate(sentiments_by_chapter, start=1):
            out.write(f"CHAPTER {chapter_num}\n")
            for sentiment in sentiments:
                out.write(str(sentiment) + "\n")
            out.write("\n")  # Separate chapters with an empty line for clarity

# Define the chapter pattern
chapter_pattern = re.compile(r'(CHAPTER \d+)(.*?)(?=CHAPTER \d+|$)', re.S)

# Read the content of the file
with open(r"path showing the original text", "r", encoding="utf-8") as file:
    content = file.read()

# Split the content roughly in half
half_length = len(content) // 2
first_half = content[:half_length]
second_half = content[half_length:]

# Extract chapters from each half using the regex pattern
first_half_chapters = chapter_pattern.findall(first_half)
```

```python
second_half_chapters = chapter_pattern.findall(second_half)

# Combine the extracted chapters from both halves
chapters = first_half_chapters + second_half_chapters

# Lists to store sentiment scores chapter-wise
all_sentences_sentiments_by_chapter = []
baoyu_sentences_sentiments_by_chapter = []
daiyu_sentences_sentiments_by_chapter = []

# Process each chapter
for chapter in chapters:
    chapter_header, chapter_content = chapter[0], chapter[1]
    # Tokenize the chapter content into sentences
    sentences = tokenize_chinese_text(chapter_content)
    # Extract sentences that mention the characters "宝玉" and "黛玉"
    baoyu_sentences = extract_sentences_with_character(sentences, "宝玉")
    daiyu_sentences = extract_sentences_with_character(sentences, "黛玉")
    # Calculate sentiment scores for all sentences, and sentences mentioning "宝玉" and "黛玉"
    all_chapter_sentiments = [calculate_sentiment(sentence) for sentence in sentences]
    baoyu_chapter_sentiments = [calculate_sentiment(sentence) for sentence in baoyu_sentences]
    daiyu_chapter_sentiments = [calculate_sentiment(sentence) for sentence in daiyu_sentences]
    # Append chapter sentiments to respective lists
    all_sentences_sentiments_by_chapter.append(all_chapter_sentiments)
    baoyu_sentences_sentiments_by_chapter.append(baoyu_chapter_sentiments)
    daiyu_sentences_sentiments_by_chapter.append(daiyu_chapter_sentiments)

# Save the results to output files
save_sentiments_to_file(r"path for all sentences", all_sentences_sentiments_by_chapter)
save_sentiments_to_file(r"path for boayu sentences ", baoyu_sentences_sentiments_by_chapter)
save_sentiments_to_file(r"path for daiyu sentences ", daiyu_sentences_sentiments_by_chapter)
```